SnapperGPS: Algorithms for Energy-Efficient Low-Cost Location Estimation Using GNSS Signal Snapshots

Jonas Beuchert
University of Oxford
Oxford, UK
beuchert@robots.ox.ac.uk

Alex Rogers
University of Oxford
Oxford, UK
alex.rogers@cs.ox.ac.uk

ABSTRACT

Snapshot GNSS is a more energy-efficient approach to location estimation than traditional GNSS positioning methods. This is beneficial for applications with long deployments on battery such as wildlife tracking. However, only a few snapshot GNSS implementations have been presented so far and all have disadvantages. Most significantly, they typically require the GNSS signals to be captured with a certain minimum resolution, which demands complex receiver hardware capable of capturing multi-bit data at sampling rates of 16 MHz and more. By contrast, we develop fast algorithms that reliably estimate locations from twelve-millisecond signals that are sampled at just 4 MHz and quantised with only a single bit per sample. This allows us to build a snapshot receiver at an unmatched low cost of $14, which can acquire one position per hour for a year. On a challenging public dataset with thousands of snapshots from real-world scenarios, our system achieves 97% reliability and 11 m median accuracy, comparable to existing solutions with more complex and expensive hardware and higher energy consumption. We provide an open implementation of the algorithms as well as a public web service for cloud-based location estimation from low-quality GNSS signal snapshots.

CCS CONCEPTS

• **Computer systems organization → Embedded and cyber-physical systems**.

KEYWORDS

Global navigation satellite system, localisation, positioning, tracking, probabilistic signal processing, low power, low cost, open source, software/hardware co-design, cloud-offloading.

ACM Reference Format:
Jonas Beuchert and Alex Rogers. 2021. SnapperGPS: Algorithms for Energy-Efficient Low-Cost Location Estimation Using GNSS Signal Snapshots. In *The 19th ACM Conference on Embedded Networked Sensor Systems (SenSys'21), November 15–17, 2021, Coimbra, Portugal.* ACM, New York, NY, USA, 13 pages. https://doi.org/10.1145/3485730.3485931

SenSys'21, November 15–17, 2021, Coimbra, Portugal

ACM ISBN 978-1-4503-9097-2/21/11.
https://doi.org/10.1145/3485730.3485931

1 INTRODUCTION

Global navigation satellite systems (GNSS), e.g., the Global Positioning System (GPS), enable localisation of objects, humans, and animals anywhere on the Earth. For example, biologists tag wild animals with GNSS receivers to track their positions [17] and gain valuable insights into their behaviour and their responses to external forces such as climate change. However, traditional GNSS receivers have disadvantages in this application scenario. As they decode satellite signals and compute locations in real time on the device itself, their signal acquisition and processing is computational and energy intensive. If operated on a battery, their lifetime is limited and they are bulky [17]. In addition, they also require good visibility of the sky for an extended period of time during initial acquisition, which is a challenge when tracking marine animals that surface only briefly. Similar issues are evident in other application domains such as wearable fitness and low-cost logistics tracking.

To address these challenges, snapshot receivers sleep for most of the time and only wake at defined intervals to record short snapshots of GNSS signals. Rather than processing these signals and estimating locations on the device itself, these receivers digitise the raw signals and store them locally. After a receiver has been recovered, its data is transferred to another computer (e.g., in the cloud) that processes the recorded snapshots, i.e., estimates the receiver's position history. This eliminates any on-board signal processing and real-time data transmission, greatly lowering hardware complexity and energy consumption. However, the processing segment then faces the challenge of estimating locations from signal snapshots that are too short to decode signal transmission timestamps or information about the satellite position. The latter information can be substituted with data from a public database, but the timestamps must be reconstructed from the raw snapshots. For this, snapshot GNSS has to rely on the unique periodic codes that all satellites broadcast and that have period lengths of one to ten milliseconds. Specifically, it employs the *code phase*, which is the shift between the start of a new code period of a certain satellite signal and the start of a signal recording. The code phase provides some information on the signal travel time from the satellite to the receiver, which can be exploited in two different ways. The first approach comprises two steps that are called *satellite acquisition*, which estimates code phases from the raw snapshot, and *coarse-time navigation* (CTN), which is a code-phase-based positioning algorithm [29]. In contrast, the second approach, which is known as *direct position estimation* (DPE), *direct positioning*, or *collective detection* (CD), has only a single step. It directly estimates the position that maximises the likelihood of observing the captured raw signal [7].

A few snapshot GPS implementations have been presented in the literature so far [3, 5, 10, 16, 28, 32]. All rely either on satellite

acquisition and CTN or on DPE/CD. They are reported to enable localisation with mean or median errors between 5 and 16 m, which are acceptable for applications such as wildlife tracking that do not require metre-level accuracy. However, several aspects are missing in the existing work on snapshot GNSS:

- Only a small number of dedicated snapshot receivers has been presented, namely CLEO built for the CO-GPS project, the solutions of Baseband Technologies and the ETH Zürich, and the ATS G10 Ultralite GPS [4, 9, 16, 18]. However, CO-GPS and the ETHZ evaluate their positioning algorithms on data from commercial GPS front-ends rather than from their developed snapshot receivers [5, 16], leaving the question open how the simplified hardware impacts the algorithm performance. Baseband Technologies do not describe the evaluation of their closed commercial offering at all.
- Existing implementations require relatively high sampling frequencies of at least 8 MHz (often 16) and several bits for the amplitude quantisation when digitising the captured snapshots [9, 10, 16], both of which increase the requirements for the on-board hardware. These are the main hurdles to overcome on the way to snapshot GNSS at low costs.
- Most systems rely on basic CTN for positioning [10, 16, 28, 32], which is not robust and prohibits using very low sampling rates and amplitude resolutions. The alternative DPE/CD algorithms are expected to be more robust, but with execution times of several seconds or minutes for a single fix too slow to efficiently calculate positions for batches with thousands of snapshots [3, 5, 7].
- Only the ATS G10 device was evaluated for wildlife tracking, but found to be unreliable due to battery and software failures [17]. For all other solutions, no performance evaluation in a real-world application scenario has been published; only simulations or stationary receivers have been considered; although, the small, light, and energy-efficient snapshot GNSS receivers are especially beneficial in dynamic scenarios.
- It is unclear how existing snapshot GNSS algorithms compare performance-wise since they have never been evaluated on a common dataset.
- No published snapshot algorithm uses multiple satellite systems. Thus, they discard available information that could aid localisation; especially, when sky visibility is limited.

The SnapperGPS project addresses all of these points. Its overall goal is to develop open-source receiver hardware and cloud-based software for energy-efficient low-cost snapshot GNSS. A core challenge is to design accurate, robust, and fast positioning algorithms that work with captured GNSS signals that:

- are as short as a few milliseconds,
- are sampled at a low rate,
- have a low amplitude resolution, and
- are noisy due to low-cost hardware components.

The availability of such algorithms would lower the requirements for receiver hardware and reduce the amount of data that it must capture and store. In the light of this challenge, this work presents a novel snapshot positioning algorithm as well as improvements of existing ones that significantly increase accuracy and reliability for low-cost receivers with coarse signal resolution using probabilistic signal processing. Specifically, we describe three conceptually different approaches to snapshot-based positioning:

(1) The first one belongs to the category of two-step approaches and comprises a tailored acquisition stage and a CTN stage based on least-squares. Our main novel contributions are Bayesian strategies to efficiently and robustly identify and remove the outliers in the code phases that the acquisition stage provides. This is essential since the low resolution and brevity of the signal snapshots causes significant numbers of outliers and we want to avoid a conservative selection strategy that erroneously discards inliers. We also present an approach to handle the considerable frequency offsets of low-cost RF front-ends that minimises computation time.
(2) The second approach relies on the same acquisition stage, but uses a novel robust maximum-likelihood estimation (MLE) instead of least-squares optimisation to jointly solve the outlier-detection problem and the final positioning problem.
(3) The third approach is a modification of state-of-the-art DPE [5], which is based on a more plausible probabilistic model, tailored to low-resolution signals, faster for snapshots longer than 1 ms, and takes into account multiple GNSS.

We investigate these three concepts to determine which one is best suited for ultra-low resolution signals (4 MHz sampling rate and one-bit amplitude quantisation) and provide implementations of all of them as the first open-source snapshot GNSS algorithm compilation[1]. In a real-world evaluation of the algorithms with moving low-cost receivers in diverse environments, we show that our outlier detection methods improve the success rate of CTN from 81% to 97% and reduce the median error from 50 m to 12 m, while increasing the algorithm runtime by only a few percent, depending on which specific outlier detection we employ. This accuracy is sufficient for many applications, e.g., wildlife tracking, and our novel MLE has an even better one of 11 m. Its runtime is three-times higher, but still low. Finally, our DPE algorithm achieves the same reliability of 97% while using less satellites, but at magnitudes higher runtimes. However, this is still an improvement over state-of-the-art DPE [5], which can only calculate fixes for 24% of our data. In general, our evaluation is the first performance comparison of different snapshot GNSS algorithms on the same data, which we make available to the public as the first open collection of thousands of GNSS signal snapshots[2]. Together with our purpose-built snapshot receiver, the novel algorithms form a snapshot-positioning system with an unmatched trade-off between costs ($14-receiver), energy consumption (15 mAh/year), and performance, which is accessible via a public snapshot GNSS online service[3].

The rest of the paper is structured as follows: Section 2 presents our hardware, which defines the constraints for our algorithms. After covering the basics of GNSS and snapshot GNSS in Sec. 3, we describe those algorithms and their evaluation in Sec. 4 and Sec. 5, respectively. Finally, Sec. 6 summarises the implementation of the algorithms in our online service before Sec. 7 provides conclusions.

[1] https://github.com/JonasBchrt/snapshot-gnss-algorithms
[2] https://doi.org/10.5287/bodleian:eXrp1xydM
[3] https://snapper-gps.herokuapp.com/

2 HARDWARE

The core components of our purpose-built SnapperGPS receiver are an SE4150L integrated GPS receiver circuit, a Silicon Labs EFM32HG310F64 microcontroller with an ARM Cortex-M0+ core and a USB interface, and a 512 MBit serial NAND flash memory IC. The bill of materials is about $14 for a unit in a batch of 100, which is significantly cheaper than any available device that is capable of performing a comparable task. The receiver must be combined with custom battery and antenna, e.g., an LP401528 lithium polymer battery adds $5 and a small Siretta Echo 27 antenna $10.

The microcontroller sleeps in a low-power mode whenever it does not record a snapshot, which consumes 12.6 mAh per year. Capturing 9000 twelve-millisecond snapshot consumes 2.5 mAh. This totals to a consumption of 15.1 mAh per year if we consider hourly fixes. In consequence, a small 110 mAh battery enables a theoretical operation time of more than seven years.

The digital signals that a SnapperGPS board stores have a length of 12 ms, are sampled at a low 4.092 MHz rate (a quarter of the receiver IC clock frequency), and have the smallest possible amplitude quantisation of one bit. This means that the signal resolution is significantly lower than what has been used for existing snapshot GPS algorithms, cf. Sec. 1, which allows the device to have less on-board memory. In addition, by operating the microcontroller on the same clock as the receiver IC, it can capture the one-bit receiver output using the existing device USART configured as an SPI master. In contrast, the solution of the ETHZ demands for two bits being sampled at 16 MHz, which limits their microcontroller choice to one with a parallel input capture interface (PARC) [9], while CLEO uses additional circuitry to convert its two-bit GPS input stream at 16 MHz into a 16-bit parallel signal at 2 MHz, which a microcontroller then captures using direct memory access (DMA) [16].

3 BACKGROUND

Four navigation satellite systems provide global coverage: the United States' Global Positioning System (GPS), Russia's GLObal NAvigation Satellite System (GLONASS), China's BeiDou Navigation Satellite System (BDS), and the European Union's Galileo positioning system. The fundamental positioning concept is the same for all GNSS [6, 27]. Multiple satellites—about 30 per system—orbit the Earth and transmit signals to the ground. A receiver picks up the signals of visible satellites and reconstructs their travel times and thus the distances from the satellites to the receiver. Then it determines its position from those distances and the satellite orbits.

3.1 GNSS Signals

Navigation satellites transmit signals in different radio frequency bands [6, 27]. Most civilian low-cost receivers operate in the GPS L1 band with a centre frequency of 1.57542 GHz which also includes the Galileo E1 signal, the BeiDou B1C signal, the novel GPS L1C signal, and the potential future GLONASS L1OCM signal, such that they can all be captured together.

All GPS satellites transmit L1 signals at the same time. These signals consist of three sub-signals, which are modulated on top of each other [6, 14, 27]:

- the sinusoidal carrier wave with a frequency of 1.57542 GHz,
- the coarse/acquisition (C/A) code, a binary signal that is unique for each satellite, has a bitrate of 1.023 MHz, repeats every millisecond, and allows to distinguish the signals of the individual satellites, and
- the binary data signal, which encodes the time when the signal was transmitted, the ephemeris describing the current satellite orbit, and the almanac, which provides information on, e.g., the states of the satellites and the ionosphere, a layer of the atmosphere that affects the signal propagation speed. The data signal's bitrate is 50 Hz and it takes 30 s to transmit a full ephemeris record, which is done every 30 s, i.e., continuously. A timestamp is transmitted every 6 s.

The E1, B1C, L1C, and L1OCM signals have a similar, but more sophisticated structure. A difference is that their codes are complex with a (real) data channel and an (imaginary) pilot channel, and the data signal is only modulated on the data channel. Furthermore, their codes do not have a duration of 1 ms. Instead, they repeat every 4 ms (E1) or 10 ms (B1C and L1C).

3.2 GNSS Receivers

The signal capturing process of a snapshot receiver is the same as the one of a conventional software-defined GNSS receiver (SDR). First, an antenna captures the raw signal. The carrier frequencies of the desired GNSS signals lay in the interval 1.57542 GHz ± 4.2 kHz [16, 27]. The deviation of up to 4.2 kHz from the centre frequency of the L1 band depends on the relative receiver-satellite velocity and is due to the Doppler effect.

Since sampling and storing data at a rate of multiple gigahertz is not possible with a low-cost receiver, an analogue circuit downconverts the frequency of the incoming signal by mixing (i.e., multiplication) with a locally generated sine wave with a frequency close to the centre frequency of 1.57542 GHz [6, 27]. If the frequency of the local sine wave is exactly the same as the frequency of the incoming signal, then this multiplication results, according to the double-angle formulae, in a signal with a DC component and a component with twice the original frequency. If the incoming and the local frequency do not coincide perfectly, then the resulting signal does not have a DC component, but one with a low frequency that is the difference between the frequency of the locally generated sine and the one of the incoming signal. The difference between the centre frequency and the receiver frequency is called the intermediate frequency (IF). After mixing, a bandpass filter removes the high frequency component such that only the near-DC component remains. Many receivers also split the incoming signal and independently mix it with sine and cosine waves such that two signals result: an in-phase (I) and a quadrature (Q) signal, which can be interpreted as real and imaginary part of a complex IQ signal.

Finally, an analogue-to-digital converter (ADC) digitises the so-called baseband signal. The ADC parameters are key for the trade-off between hardware costs, maximum operation time, and positioning performance. A relatively small sampling rate lowers the hardware requirements and reduces the amount of data to store. A small number of quantisation intervals, i.e., a coarse amplitude resolution, has the same effect. However, both choices also lower the quality of the digitised signal, and, thus, render subsequent signal processing harder.

3.3 Satellite Acquisition

The goal of satellite acquisition is to identify which satellite's signals are present in the captured snapshot and to determine their code phases. A popular acquisition algorithm is parallel code phase search (PCPS) [1, 6, 27]. First, it generates replicas of the codes of the satellites that are expected to be visible from a coarse initial position. It then generates carrier wave replicas for the signals of the same satellites. The frequencies of these complex sinusoidal waves are the sums of the nominal IF, the unique frequency offset of the specific RF front-end, and the expected Doppler shifts that results from the relative satellite speeds. If one of these terms is uncertain, then PCPS generates multiple carrier wave replicas with different frequencies that span the desired frequency search space for each satellite. It then individually multiplies the incoming signal with the carrier wave replicas to down-shift the signal to a lower frequency, similar to the mixing procedure of the analogue front-end circuit. If the frequency of the generated signal and the one of the incoming signal from the respective satellite coincidence, then the carrier wave is effectively removed and only code and data signal remain. The resulting signals are then correlated with the respective code replicas to determine their similarities. To reduce the computation time, this is done in the frequency domain after fast Fourier transformation (FFT). Since the codes of the individual satellites are almost orthogonal to each other, the correlation is little influenced by the presence of signals from other satellites in the snapshot. Furthermore, the codes have an auto-correlation function with a single narrow peak[4] such that a high correlation between a code replica and the incoming signal is only expected at a lag close to zero and almost zero correlation is expected elsewhere. Finally, the resulting correlograms are searched for the highest peak. For each satellite, the location of this peak is presumed to correspond to the code phase. However, the peak could be caused by noise or a satellite signal that did not arrive via a direct line of sight, i.e., by reflections. Therefore, a measure for the reliability that the estimated code phase corresponds to a directly received GNSS signal is needed, e.g., the signal-to-noise ratio (SNR), the value of the highest correlogram peak [6], or the ratio between the highest and the second highest peak [16].

The PCPS procedure opens up a few design choices. For example, there are multiple ways to process snapshots that are longer than the code period of the satellite signal. The first option, *coherent integration*, repeats the satellite codes such that they have the same length as the snapshot and calculates the correlation over the full incoming signal and the sequences of repeated codes. In contrast, the second option, *non-coherent integration*, splits the incoming signals into chunks whose length equals the code length. It then individually calculates the correlation for each signal chunk and a single code period. Finally, it sums the correlograms of the individual chunks of each whole snapshot before it searches for the peak. Both strategies account only for the carrier wave and the code signal and not for the third GNSS signal component, the binary navigation data. Since the bitrate of the navigation data is much smaller than the one of the codes, it effectively either maintains or flips the sign of all code bits in one code period. Fortunately, a persistent flip of the sign of one of the operands of a correlation operation does not change its outcome. Therefore, neglecting the navigation data signal does not have an impact for signal chunks that are as short as the code period. However, coherent integration is not robust to a change of the data signal during the snapshot. If the data bit flips in the middle of the snapshot, then the contributions of the first signal half and of the second signal half effectively cancel out, causing a correlation of approximately zero. Non-coherent integration does not have this problem since it individually calculates the correlations of the signal chunks [16]. Furthermore, the longer the processed signal chunk is, the more precisely the carrier frequency must be known. For one-millisecond chunks, the frequency should be known down to 500 Hz, and for ten-millisecond chunks, down to 50 Hz [27]. It is also possible to choose an option between both integration types by selecting a chunk length for non-coherent integration that is a multiple of the code period, but still smaller than the signal length.

For signals with two components, e.g., E1, B1C, and L1C, there is a second design choice w.r.t. combining the information from both channels [23, 24, 31]. Of course, it is possible to use just the pilot or just the data channel for acquisition, but this will discard information and decrease sensitivity. The first technique, the *coherent* one, calculates the correlation for the sum of the two replica channels. However, because the value of the navigation data bit is unknown, it is possible that it is negative and the code in the data channel is actually inverted. To account for this, the coherent technique calculates the correlation for the difference of both replica components, too, and chooses the correlogram with the maximum power to estimate the code phase. The second technique, the *non-coherent* one, individually calculates the correlations for both channels and sums the resulting correlograms before searching for the peak.

3.4 Coarse-Time Navigation

Coarse-time navigation uses the code phases of the acquired satellites to estimate the receiver position [29]. A challenge for CTN in contrast to standard positioning is that the code phases are not measurements of the full signal travel times between the satellites and the receiver. Instead, they correspond to the signal travel times modulo the code period, e.g., for GPS L1, the code phases are the sub-millisecond parts of the signal travel times because the L1 codes have a period of 1 ms. Therefore, CTN firstly reconstructs the full travel times. Based on an initial coarse estimate of receiver position and time and the satellite ephemeris, it estimates a coarse satellite-receiver distance and rounds the corresponding signal travel time to the next smaller multiple of the code period. The full reconstructed signal travel time is then the sum of this value and the code phase. To ensure the consistency of the reconstructed times of all satellites, the satellite that is expected to be most reliable can be chosen as reference and the coarse travel times of the signals of the other satellites calculated relative to the coarse signal travel time of the reference satellite's signal [29]. For a correct reconstruction, the error of the coarse position should not be larger than the distance that the signal travels during half a code period. Similarly, the error of the time should be small enough to ensure that the satellite has not travelled by more than the same distance. In practice, CTN for GPS tolerates position errors of 100–150 km and time errors of up to 1 min [29].

[4] E1, B1C, and L1C have two additional smaller peaks.

The signal travel times are divided by the speed of light to obtain the so-called pseudoranges, which correspond to the hypothetical satellite-receiver distances if the signals would travel with exactly the speed of light and signal emission and reception would happen without any delays, which is both not the case in practice. For a given receiver position $\mathbf{p}_\mathrm{r} \in \mathbb{R}^3$ and timestamp $t_\mathrm{r} \in \mathbb{R}$, a pseudorange $\rho \in \mathbb{R}^+$ can be more accurately predicted with

$$\rho = \|\mathbf{p}_\mathrm{s}\,(t_\mathrm{r} + \delta t_\mathrm{r}) - \mathbf{p}_\mathrm{r}\| + b - c \cdot \delta t_\mathrm{s}\,(t_\mathrm{r} + \delta t_\mathrm{r}) + a\,(t_\mathrm{r} + \delta t_\mathrm{r}, \mathbf{p}_\mathrm{r})\,. \quad (1)$$

In addition to the precise receiver position $\mathbf{p}_\mathrm{r}$, this equation contains two further unknown variables: first, the coarse receiver time error $\delta t_\mathrm{r} \in \mathbb{R}$, which accounts for an imprecise timestamp t_r, and, second, a common bias $b \in \mathbb{R}$ in the pseudoranges to all satellites, which is caused by, e.g., common delays in the signal capturing process and the same for all satellites. The satellite position $\mathbf{p}_\mathrm{s}\colon \mathbb{R} \to \mathbb{R}^3$ and the satellite clock correction $\delta t_\mathrm{s}\colon \mathbb{R} \to \mathbb{R}$ are calculated from the satellite's navigation data. The latter accounts for a known error of the satellite's internal clock, which is mapped to an equivalent distance by multiplication with the speed of light c. An atmospheric model $a\colon \mathbb{R} \times \mathbb{R}^3 \to \mathbb{R}^+$ accounts for a signal delay during the propagation in the ionosphere and troposphere [6, 12, 13, 25].

Coarse-time navigation uses non-linear least-squares optimisation, i.e., repeated linearisation followed by standard linear least-squares optimisation, to estimate the receiver position, the common bias, and the coarse time error that minimise the sum of the quadratic errors between the reconstructed pseudoranges and their corresponding predictions. Weighting of the individual observations is possible, if desired. The least-squares method provides an uncertainty estimate of the solution, too, which consists of a factor for the uncertainty of the observed pseudoranges and a factor that is driven by the geometry of the identified satellites, the dilution of precision (DOP) [21, 22, 29]. The main drawback of CTN is that it is not robust to incorrectly estimated code phases. A wrong code phase estimate can cause an error of the reconstructed pseudorange of more than 100 km. Such a single outlier is very likely to prevent the least-squares method from succeeding. On the other hand, it is a fast algorithms with reported runtimes of 0.1–0.5 s, including GPS satellite acquisition [4, 10, 16].

3.5 Direct Positioning

Direct positioning, which is also known as direct position estimation (DPE) or collective detection (CD), does not separate satellite acquisition from position estimation and aims at robustifying snapshot GPS for challenging signals [3, 5, 7]. It is based on a probabilistic model that maps a receiver position and time hypothesis to a set of satellites, which are expected to be visible, and their transmitted signals. Given these expected signals, DPE/CD calculates the likelihood of observing the captured signal and maximises it over a set of hypotheses. For the latter, Closas Gómez proposes different methods that optimise over the four-dimensional space spanned by receiver position and time [7]. All methods lead to accurate positioning in simulations, but two problems occur in practice: First, the optimisation does not account for a common bias, a delay that cannot be avoided when using real hardware and must be treated as an additional unknown variable, cf. Sec. 3.4. Second, all methods are computationally expensive and optimisation over search spaces with diameters larger than a few hundred metres is infeasible.

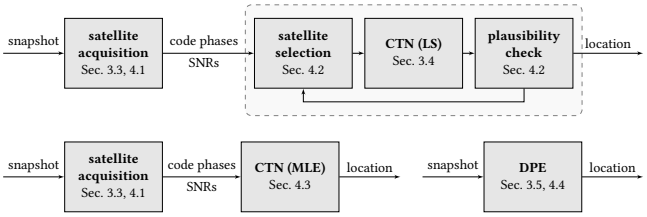

Figure 1: Block diagrams of SnapperGPS' alternative positioning algorithms: *LS-linear*, *LS-combo*, and *LS-SAC* (top), *MLE* (bottom left), and *DPE-[5]* and *DPE+* (bottom right).

Bissig et al. present an accelerated optimisation algorithm for one-millisecond snapshots, which enables them to generously bound the temporal search space to 10 s and the spatial search space to up to 200 km x 200 km x 30 km while keeping the computation time down to seconds or minutes [5]. For this, they discretise the search space, which allows brute-force grid optimisation of the common bias and efficient branch-and-bound optimisation over a set of four-dimensional hypotheses consisting of receiver position and time. To handle a snapshot that is longer than one millisecond, they spilt it into one-millisecond chunks that they process individually. Finally, they average over the results to obtain a solution for the whole snapshot. Despite its hierarchical structure and precomputations before the start of the optimisation, the algorithm's runtime is still several magnitudes higher than the one of acquisition and CTN combined, especially for signals with a duration of multiple milliseconds. Bissig et al. claim that it can be implemented on a GPU for a significant speed-up, but it is not clear how this could be achieved since branch-and-bound requires sequential processing of hypotheses but a GPU requires large-scale parallelisation to decrease computation time.

4 ROBUST ALGORITHMS

SnapperGPS implements eight alternative algorithms for positioning based on GNSS snapshots, cf. Fig. 1. ***LS-linear*** is a traditional approach with satellite acquisition and non-linear least-squares optimisation and ***DPE-[5]*** an implementation of Bissig et al.'s DPE, both of which are provided as baselines. The first three novel algorithms ***LS-linear***, ***LS-combo***, and ***LS-SAC*** combine the tailored acquisition stage from Sec. 4.1 with non-linear least-squares optimisation for positioning. They employ different strategies to select the satellites with reliable code-phase observations, which we present in Sec. 4.2 and which have different trade-offs between robustness and runtime. Furthermore, Sec. 4.3 introduces ***MLE***, our novel maximum-likelihood approach to snapshot-positioning, and ***LS-SAC/MLE***, a combination of *LS-SAC* and *MLE*. Both aim at higher accuracy and robustness at only slightly increased computational costs. Finally, we summarise our direct positioning algorithm ***DPE+***, which is tailored to low-resolution signals, too, in Sec. 4.4.

4.1 Snapshot-Based Satellite Acquisition

The three main requirements for our tailored acquisition stage are (I) to minimise computation time and (II) to achieve high reliability and (III) accuracy of the code-phase estimates, although, the signal

resolution is low. The first step towards (I) is to minimise the search space, which has three dimensions: satellite index, carrier frequency, and code phase, cf. Sec. 3.3. To reduce the number of satellites to search for, we predict the elevation angle above the horizon for all GPS, Galileo, and BeiDou satellites using their ephemerides and an initial coarse estimate of receiver position and time. Then we only consider those whose elevation is above a threshold, i.e., those which are more likely to have a line of sight towards the receiver.

Furthermore, we predict the Doppler shifts for the selected satellites and only search the frequency dimension around them.

Third, we estimate the potentially large frequency offset of each low-cost receiver front-end at the beginning of a recording with a new receiver and correct the IF accordingly. If this estimate is accurate, then we can skip the search along the frequency axis for subsequent snapshots from the same device. We propose two alternative algorithms to estimate the frequency offset of the RF front-end. Both require us to carry out full acquisitions with enlarged frequency search spaces for the first few snapshots of a data record. Therefore, we describe them at the end of this subsection 4.1.

We also expect an amplitude offset of the raw snapshot due to the low-cost ADC. To avoid a DC component in the spectrum, we subtract the mean of each snapshot before executing the acquisition.

For the acquisition itself, we split the twelve-millisecond snapshots into individual chunks whose lengths equal the code period of the considered GNSS signal, apply zero padding if necessary, and use non-coherent integration across all signal chunks and all channels, if there is more than one. Non-coherent integration is more robust than coherent integration, cf. Sec. 3.3, an important feature for signals with low resolution, which addresses requirement (II).

If we have access to a GPU, we employ it to perform the FFTs and inverse FFTs for the correlation with NVIDIA's cuFFT library. For this, we parallelise the acquisition across all satellites, the whole frequency search space, and as many snapshots as possible to use the full memory of the GPU and minimise the number of data transfers from and to the CPU. This is another step towards requirement (I).

For requirement (III), it is important that PCPS as described in Sec. 3.3 considers the index of the highest correlogram peak to correspond to the code phase. However, this limits the resolution of the code-phase estimates by the size of the sampling intervals. If the sampling frequency is low, then the estimates are imprecise, which leads to large errors of the reconstructed receiver-satellite distances. For example, a sampling frequency of 4.092 MHz corresponds to a distance resolution of about 73 m. The acquisition stage of traditional GNSS receivers does not need to provide precise code phases because they are refined in the subsequent tracking stage that operates on much longer signal streams. However, this is not possible if only short signal snapshots are available. Therefore, we apply quadratic interpolation around the correlogram peak to increase the code-phase resolution, a procedure that is already known to be successful in the tracking stage of the traditional processing chain [27]. We fit a quadratic function to a few points around the peak and consider the maximum of the quadratic function to be at the true code phase.

The procedure described so far provides code-phase estimates for any satellites that could potentially be visible to the receiver. However, some of the GNSS signals might be blocked by, e.g., surrounding vegetation or infrastructure. Therefore, we are interested in a measure that can be used to determine whether a code phase corresponds to a received GNSS signal or occurs just due to noise, i.e., is an outlier, cf. Sec. 3.3. We pick the SNR for this measure and calculate it for each code-phase estimate by considering the correlogram peak as the *signal* and the remaining correlogram after removing the peak and a few samples on both sides of it as the *noise* and dividing both normalised values.

Having introduced all other steps of the acquisition, we can go back to the two alternative frequency offset estimation algorithms. One of both is executed after acquisition runs with enlarged frequency search spaces for the first few snapshots of a data record.

The first algorithm simply calculates the differences between the carrier frequency estimates that are returned by the acquisition and the ones that we predicted. Then, it takes the median of these differences across several satellites and snapshots. However, this approach is less robust if the IF is close to zero. E.g., if the IF is exactly zero, then a carrier frequency that differs by -50 Hz from the IF is indistinguishable from one that deviates by +50 Hz.

We solve this problem using a probabilistic model and MLE. At first, we derive a prior probability $P\left(v_i = 1|\mathrm{SNR}_i\right)$ for each satellite observation $i \in 1,\dots N$ to be reliable, i.e., to be a so-called inlier, given the associated SNR. We model the distribution $p\left(\mathrm{SNR}_i\right)$ of the SNRs as a Gaussian mixture model with two components, $p\left(\mathrm{SNR}_i|v_i = 1\right)$ for the inliers and $p\left(\mathrm{SNR}_i|v_i = 0\right)$ for the outliers. We fit mean, standard deviation, and prior of each component to a training dataset. We do this separately for each GNSS since the GPS L1 signal, the Galileo E1 signal, and the BeiDou B1C signal have different properties and, therefore, differently distributed SNRs. Using the resulting probabilistic models and Bayes' rule, we obtain a prior $P\left(v_i = 1|\mathrm{SNR}_i\right) = \frac{p\left(\mathrm{SNR}_i|v_i=1\right)P\left(v_i=1\right)}{p\left(\mathrm{SNR}_i\right)}$ for each satellite to be an inlier and $P\left(v_i = 0|\mathrm{SNR}_i\right) = 1 - P\left(v_i = 1|\mathrm{SNR}_i\right)$ to be an outlier.

Next, we start Alg. 1 and predict the carrier frequency $\hat{f}_i \in \mathbb{R}$ for all satellites in line 2 based on the nominal IF f_{IF} and the expected Doppler shift of the L1 centre frequency f_{L1} caused by the time-dependent satellite velocity $\mathbf{v}_{\mathrm{s},i} \colon \mathbb{R} \to \mathbb{R}^3$. For each prediction, we get the observed carrier frequency $f_i \in [f_{\min}, f_{\max}]$ from the initial acquisition runs over the enlarged search space $[f_{\min}, f_{\max}]$. To account for the ambiguity of the sign $s_i \in \{-1, +1\}$ mentioned above, we create two instances of each observation, one with the original sign $s_i = +1$ and one with the inverted sign $s_i = -1$ and assign half the prior probability to each value in line 4. We assume all observations to be Gaussian distributed if they are inliers and all outliers to be sampled from a uniform distribution over the bandwidth of the frequency search space, cf. lines 5–6. Next, we marginalise the sign s_i and the validity v_i in lines 8–11 and join the individual satellites in line 13.

We assume an unknown common offset δf_{r} in all observations, cf. line 2, which we estimate by maximising the output of Alg. 1, i.e., the likelihood $p\left(f_1,\dots f_N|\mathbf{p}_{\mathrm{r}}, \delta f_{\mathrm{r}}, t_{\mathrm{r}}, \mathrm{SNR}_1,\dots \mathrm{SNR}_N\right)$, over δf_{r} in a constrained search space. In fact, we minimise the corresponding negative log-likelihood with a limited-memory quasi-Newton method that approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimisation algorithm. We repeat the optimisation several times with decreasing assumed standard deviations $\sigma_{\mathrm{f}} \in \mathbb{R}^+$ of the frequency observations to refine the solution and initialise each iteration with the result from the previous one.

Algorithm 1 Likelihood for observing a set of carrier frequencies given receiver position, frequency offset, time, and SNRs

Procedure: $p\left(f_1, \dots f_N | \mathbf{p}_\mathrm{r}, \delta f_\mathrm{r}, t_\mathrm{r}, \mathrm{SNR}_1, \dots \mathrm{SNR}_N\right)$

for $i \in \{1, \dots N\}$ **do**
  $\hat{f}_i \leftarrow f_\mathrm{IF} - \frac{\left(\mathbf{v}_{\mathrm{s},i}(t_\mathrm{r})\right)^\mathrm{T}\left(\mathbf{p}_{\mathrm{s},i}(t_\mathrm{r}) - \mathbf{p}_\mathrm{r}\right)}{\left\|\mathbf{p}_{\mathrm{s},i}(t_\mathrm{r}) - \mathbf{p}_\mathrm{r}\right\|} \frac{f_\mathrm{L1}}{c} + \delta f_\mathrm{r}$
  for $\hat{s} \in \{-1, +1\}$ **do**
    $p\left(s_i = \hat{s}\right) \leftarrow \frac{1}{2}$
    $p\left(f_i | \hat{f}_i, v_i = 1, s_i = \hat{s}\right) \leftarrow \frac{1}{\sqrt{2\pi}\sigma_\mathrm{f}} \exp\left(\frac{\left(\hat{s} f_i - \hat{f}_i\right)^2}{\sigma_\mathrm{f}^2}\right)$
    $p\left(f_i | \hat{f}_i, v_i = 0, s_i = \hat{s}\right) \leftarrow \frac{1}{f_\mathrm{max} - f_\mathrm{min}}$
  end for
  for $\hat{v} \in \{0, 1\}$ **do**
    $p\left(f_i | \hat{f}_i, v_i = \hat{v}\right) \leftarrow \sum_{\hat{s} \in \{-1, +1\}} p\left(f_i | \hat{f}_i, v_i = \hat{v}, s_i = \hat{s}\right) p\left(s_i = \hat{s}\right)$
  end for
  $p\left(f_i | \hat{f}_i, \mathrm{SNR}_i\right) \leftarrow \sum_{\hat{v} \in \{0,1\}} p\left(f_i | \hat{f}_i, v_i = \hat{v}\right) P\left(v_i = \hat{v} | \mathrm{SNR}_i\right)$
end for
return $\prod_{i=1}^{N} p\left(f_i | \hat{f}_i, \mathrm{SNR}_i\right)$

4.2 Satellite Selection for Coarse-Time Navigation

The SNR-induced reliability measure that we use at the end of Sec. 4.1 to rate the observed carrier frequencies can also be used to assess the reliability of the second type of output of the acquisition stage, the code-phase estimates. A straight-forward next step would be to either employ all satellites with a prior probability $P\left(v_i = 1 | \mathrm{SNR}_i\right)$ above a pre-defined threshold or a fixed number of satellites with the highest priors for a single CTN via non-linear least-squares as in Sec. 3.4. We refer to the latter method as *LS-single* in the following. However, choosing an appropriate threshold or satellite count is not trivial and there is always the threat that we either discard a satellite that is in fact reliable or select a non-reliable satellite for the position calculation. The latter case will almost certainly cause the least-squares routine to fail since it is not robust to outliers, cf. Sec. 3.4. On the other hand, the first case is undesired, too, because more observations usually lead to improved accuracy since errors average out and the confidence in the result increases. In consequence, there is the need for intelligent algorithms for satellite selection. Our proposals rely on iterative solution checking. The simplest procedure (*LS-linear*) executes the least-squares CTN for a large number of satellites, whose observations are more likely to be inliers given the SNR. Next, we check if the solution is plausible. Examples for plausibility checks are

- spatial and temporal proximity of the obtained solution w.r.t. a previous or initial fix,
- a reasonable height estimate close to the surface of the Earth,
- small residuals for all observations,
- a small uncertainty associated with the solution, and
- a minimum number of additional observations that are not in the selected subset is predicted approximately correctly.

If the solution is not plausible, then we remove the satellite that is least likely to be reliable from the set and apply the least-squares algorithm again. If the solution is still not plausible, then we remove the satellite with the second smallest prior probability, too, and repeat. In this way, we iterate until we either find a plausible solution or too less satellites are left to solve the system of equations.

The second algorithm (*LS-combo*) works similar. However, before we remove a second satellite from the set of potentially reliable satellites, we first add the satellite that has been removed first back to the set and remove just the second least likely reliable one. In this way, we iterate through all subsets of the initial set of reliable satellites. We start with the full set, continue with all subsets that are by one satellite smaller, and then with all subsets that miss two satellites, and so on. We abort this combinatorial search as soon as we find a first plausible solution. *LS-combo* has the disadvantage of a large runtime when many outliers exist since it falls into the complexity class $O(N^{\frac{N}{2}})$ where $N \in \mathbb{N}$ is the total number of available observations.

The third algorithm (*LS-SAC*) addresses this. It is inspired by the robust RANSAC optimisation used in computer vision [11], adapted to our specific problem and its non-linear character, and comprises the steps listed in Alg. 2. Its goal is to quickly select a large set of reliable satellites for CTN by testing only small subsets and then

Algorithm 2 *LS-SAC*

Inputs: code phases, SNRs, initial receiver position, coarse time.
Outputs: receiver position, common bias, coarse time error.

Define a small number of code phases that is at least large enough to solve the least-squares optimisation in theory.
Find all code-phase subsets with this number of elements.
loop
  for all subsets that have not been tested **do**
    Calculate the probability that the set contains only inliers given the SNRs and that there was most likely at least one outlier in all previously tested subsets, cf. App. A.
  end for
  Select the subset with the highest probability.
  Apply CTN via non-linear least-squares as described in Sec. 3.4 to calculate the solution given this subset.
  Check if the solution is plausible as described for *LS-linear*.
  if the solution is plausible **then**
    Predict all other code phases (that are not part of the selected subset) based on the solution.
    Determine all code phases that differ by less than a threshold from their corresponding predictions.
    Update the solution using the original small subset of code phases and those additional ones for a further least-squares step.
    return the updated solution.
  else
    if a termination criterion is fulfilled **then**
      return that the algorithm failed.
    end if
  end if
end loop

also incorporating all satellites whose code phases approximately agree with the solution from the tested subset.

Examples for termination criteria in line 16 are:

- the maximum probability of an untested subset to contain only inliers is smaller than a pre-defined threshold,
- a maximum number of iterations is reached, and
- no subset of the size defined in line 1 remains untested.

Although five observations are in theory enough to solve the least-squares optimisation problem for the five unknown variables, we start the algorithm with an initial subset size of six satellites to account for a potential unfavourable geometry of the satellite constellation. If we cannot find a plausible solution this way, then we repeat the algorithm with a subset size of five.

The probabilistic nature of *LS-SAC* allows us to calculate measures such as the expected number of iterations to be successful or the prior probability to find a plausible solution at all and pick the hyperparameters accordingly to obtain desired (maximum/minimum) values for those quantities. However, its main advantage is that it is relatively fast since most calculations are only performed for a small subset of the observations, but it is also accurate because lines 10–14 incorporate as many observations as possible for the final solution.

No matter which algorithm we use for the satellite selection, the least-squares optimisation always provides an uncertainty measurement for each solution. Specifically, we are interested in the horizontal dilution of precision (HDOP), a measure of the influence of the satellite constellation geometry on the horizontal uncertainty. If $\mathbf{H}$ is the Jacobian of the linearised model in the final optimisation step using east-north-up (ENU) coordinates, then the HDOP is [29]

$$\begin{aligned} \mathbf{Q} &= \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1} \\ \mathrm{HDOP} &= \sqrt{\mathbf{Q}_{11}+\mathbf{Q}_{22}}. \end{aligned} \tag{2}$$

We multiply the HDOP with an assumed uncertainty $\sigma_\rho \in \mathbb{R}^+$ of the observations to obtain an uncertainty measure $\hat{\sigma} = \sigma_\rho \cdot \mathrm{HDOP}$ in metres. The observation uncertainty σ_ρ is constant and pre-fit to a training dataset.

4.3 Code-Phase-Based Positioning via Maximum-Likelihood Estimation

As a robust alternative to CTN via least-squares, we introduce code-phase-based positioning via MLE. This requires two ingredients: (I) a probabilistic model that provides a likelihood $p\left(\phi_1,\dots\phi_N|\mathbf{p}_\mathrm{r},b,\delta t_\mathrm{r}\right)$ for observing the code phases $\phi_1,\dots\phi_N \in \mathbb{R}^+$ given a receiver position, common bias, and coarse time error hypothesis $(\mathbf{p}_\mathrm{r},b,\delta t_\mathrm{r})$ and (II) a method to maximise this likelihood over a set of hypotheses.

For (I), we begin Alg. 3 again with the likelihood $P\left(v_i=1|\mathrm{SNR}_i\right)$ of a satellite $i \in 1,\dots N$ to be reliable given the SNR, cf. Sec. 4.1 and 4.2. Next, we start with an initial five-dimensional hypothesis $(\mathbf{p}_\mathrm{r},b,\delta t_\mathrm{r})$ comprising the receiver position, the common bias, and the coarse time error and predict the corresponding pseudoranges $\hat{\rho}_1,\dots\hat{\rho}_N$ using Eq. (1) in line 2. However, depending on the desired accuracy, it is possible to neglect some of the terms, e.g., the atmospheric delays, to keep them constant during the whole optimisation for a single fix, or to even linearise the pseudorange prediction for a given initialisation. Subsequently, we convert the predicted full pseudoranges into code phases by dividing them by the speed of light and applying the modulo operator with respect to the code period T_i of the respective GNSS. Next, we assume that the distribution of the code-phase observations is a mixture of a Gaussian component $p\left(\phi_i|\hat{\rho}_i,v_i=1\right)$ for the valid code phases and a uniform distribution $p\left(\phi_i|\hat{\rho}_i,v_i=0\right)$ over the whole code period for the invalid code phases, cf. lines 3–4. The Gaussian is centred at the predicted code phase. We also map the difference between code-phase observation and prediction to the interval $\left[-\frac{T_i}{2},+\frac{T_i}{2}\right]$. We obtain the final likelihood $p\left(\phi_i|\hat{\rho}_i,\mathrm{SNR}_i\right)$ for observing a certain code phase by marginalising over the validity v_i in line 5.

The likelihood to observe a whole set of code phases is the product of the likelihoods for each individual observation, cf. line 7. However, we sum the logarithms of the likelihoods instead for numerical reasons and obtain the log-likelihood, which has its maximum at the same location. We maximise this function with a gradient-based local optimisation algorithm over a constraint continuous search space. The optimisation uses ENU coordinates for the receiver position to potentially constrain the search space tighter in the vertical dimension than in the two horizontal ones. Similar to the frequency offset estimation in Sec. 4.1, we perform multiple iterations in each of which we reduce the assumed standard deviation $\sigma_\rho \in \mathbb{R}^+$ of the valid observations to produce a solution with greater confidence with which we initialise the next iteration.

Algorithm 3 Likelihood for observing a set of code phases given receiver position, common bias, time, and SNRs

Procedure: $p\left(\phi_1,\dots\phi_N|\mathbf{p}_\mathrm{r},b,t_\mathrm{r}+\delta t_\mathrm{r},\mathrm{SNR}_1,\dots\mathrm{SNR}_N\right)$

1: **for** $i \in \{1,\dots N\}$ **do**

2: $\quad\hat{\rho}_i \leftarrow \left\|\mathbf{p}_{\mathrm{s},i}\left(t_\mathrm{r}+\delta t_\mathrm{r}\right)-\mathbf{p}_\mathrm{r}\right\|+b-c\delta t_{\mathrm{s},i}\left(t_\mathrm{r}+\delta t_\mathrm{r}\right)+a\left(t_\mathrm{r}+\delta t_\mathrm{r},\mathbf{p}_\mathrm{r}\right)$

3: $\quad p\left(\phi_i|\hat{\rho}_i,v_i=1\right) \leftarrow \frac{1}{\sqrt{2\pi\sigma_\rho}}\exp\left(\frac{\left(\mathrm{mod}\left(\phi_i-\frac{\hat{\rho}_i}{c}+\frac{T_i}{2},T_i\right)-\frac{T_i}{2}\right)^2}{\sigma_\rho^2}\right)$

4: $\quad p\left(\phi_i|\hat{\rho}_i,v_i=0\right) \leftarrow \frac{1}{T_i}$

5: $\quad p\left(\phi_i|\hat{\rho}_i,\mathrm{SNR}_i\right) \leftarrow \sum_{\hat{v}\in\{0,1\}} p\left(\phi_i|\hat{\rho}_i,v_i=\hat{v}\right)P\left(v_i=\hat{v}|\mathrm{SNR}_i\right)$

6: **end for**

7: **return** $\prod_{i=1}^{N} p\left(\phi_i|\hat{\rho}_i,\mathrm{SNR}_i\right)$

The MLE is designed to be robust w.r.t. outliers, but might have high runtimes if the search space is large or the initialisation bad. While we can easily obtain an initial position and an initial coarse time error from the previous fix of the track, there is no straightforward way to obtain a good initialisation for the common bias. Therefore, we introduce a modified initial MLE run over just the common bias. For the search space of the bias, we must take into account that it has a different range for different GNSS signals depending on their code period, cf. Sec. 3.1. For example, for the Galileo E1 signal it lies between zero and the distance that corresponds to 4 ms, while for the BeiDou B1C signal, the common bias can take values up the distance corresponding to 10 ms. In consequence, we constrain the search space to the interval between zero and the distance corresponding to the least common multiple of the present code periods, e.g., 20 ms if we consider E1 and B1C signals.

It does not matter that the resulting common bias can overshoot the individual code periods because we apply the modulo operator when turning pseudorange predictions into code-phase predictions. We then coarsely discretise the common bias search space and calculate the log-likelihood that we introduced above for an initial position and an initial coarse time error and all elements of the bias search space. Finally, we pick the common bias that maximises the (log-)likelihood and start with the actual five-dimensional MLE.

We denote the whole algorithm as *MLE* and also propose *LS-SAC/MLE*, a combination of *LS-SAC* and *MLE* where we apply *LS-SAC* first, then predict the uncertainty of the solution as in Sec. 4.2, and proceed with *MLE* to compute an alternative solution if and only if the predicted uncertainty exceeds a predefined threshold.

4.4 Direct Positioning

As a baseline for a one-step alternative to acquisition followed by positioning, we implement Bissig et al.'s branch-and-bound optimisation algorithm for DPE/CD (named *DPE-[5]*), which is the only existing algorithm of this class that promises robust positioning at acceptable runtimes, cf. Sec. 3.5. Our version *DPE+* introduces a few changes and new features for enhanced performance with low-quality signals. Unlike Bissig et al. [5] and more like Closas Gómez [7], we derive our likelihood function from a probabilistic model of the received GNSS signal snapshot. After applying some simplifications and the logarithm, this results in the cost function

$$\log\left(p\left(\mathbf{y}|\mathbf{p}_\mathrm{r}, b, \delta t_\mathrm{r}\right)\right) \approx \sum_{i=1}^{N} \left(\mathrm{corr}\left(\mathbf{y}, \mathbf{r}_i\left(\mathbf{p}_\mathrm{r}, b, \delta t_\mathrm{r}\right)\right)\right)^2, \qquad (3)$$

where $\mathrm{y} \in \{-1,+1\}^K$ is the captured signal and $\mathbf{r}_i : \mathbb{R}^5 \rightarrow \{-1,+1\}^K$ the replica of the signal of the i-th satellites, both with the K samples of a code period. The replicas consist of carrier wave and code and account for Doppler shift and code phase, which we predict based on the receiver position, the common bias, and the coarse time error as in line 2 of Alg. 1 and line 2 of Alg. 3. For the prediction of the code phase, we allow for pseudorange approximation or linearisation like we do for our two-step MLE in Sec. 4.3 to enable a different accuracy vs. runtime trade-off. Furthermore, we process snapshots that are longer than the code period of the GNSS signal by using non-coherent integration, i.e., by summing the correlations for the individual signal chunks with K samples. In contrast, Bissig et al. independently apply their algorithm to each chunk and average the returned positions. This is less robust and slower since the time-consuming branch-and-bound algorithm must run more often. Finally, our version does not only work with the GPS L1 but also with the Galileo E1 signal to optionally increase the number of available satellites, and, therefore, robustness and accuracy. As in Sec. 4.1, we non-coherently integrate over both E1 channels.

5 EVALUATION

To assess the algorithm performances under realistic conditions, we use large GNSS data collections with thousands of snapshots from various scenarios. Section 5.1 introduces the datasets, Sec. 5.2 describes the evaluated algorithms and the conducted experiments, and Sec. 5.3 presents the results, which we discuss in Sec. 5.4.

5.1 Data

The first data collection that we use in our experiments was recorded in 2018 and made available to the public by Watson et al.[5] [30]. It consists of three automotive driving tests in an urban environment, which contain multiple situations that degrade GNSS data, e.g., partial and total satellite occlusion through high-rise buildings or bridges. Watson et al. used a LabSat 3 GNSS signal recorder to capture the data with a sampling rate of 16.368 MHz. The signals are binary in-phase and quadrature (IQ) data with one bit per component and sample. A conventional GNSS receiver, which uses differential GNSS for accurate positioning, provides ground-truth tracks. We extract a snapshot every 10 s such that we obtain 366 in total. To render the experiments more challenging, we low-pass filter and down-sample them at 4.092 MHz, a quarter of the original rate, and use only the in-phase (I) component. With the latter alone, we expect to degrade the SNR by 1.4 dB [26, 27]. Finally, we use our robust frequency offset estimation algorithm from Sec. 4.1 for a rough estimate (±25 Hz) of the receiver front-end's frequency error and correct the IF accordingly. We obtain coarse estimates of the receiver clock errors and adjust the timestamps, too.

We established the second data collection[2] by ourselves in 2020 and 2021 using three of our SnapperGPS low-cost receivers with small Siretta Echo 27 antennas, cf. Sec. 2. It consists of four static and seven dynamic tests under various dynamically changing conditions with 3700 snapshots in total. We captured the 225 static snapshots on a hill top, on a bridge, in a courtyard, and in a park in 5–30 s intervals and the 3475 dynamic ones while cycling in either urban or rural environments and using 10 s intervals together with ground truth locations and tracks. Again, we estimate the frequency offsets of every test receiver based on five to ten snapshots from the start of a recording (until we are confident that we have estimated the offsets down to ±25 Hz) and subtract them from the nominal IF.

5.2 Experiments

Our main Experiment 1 is to apply the eight algorithms from Sec. 4 with the parametrisations in App. B to our own data collection[2] using all GNSS that broadcast signals with 1.57542 GHz centre frequency, i.e., GPS (L1 signal), Galileo (E1 signal), and BeiDou (B1C signal). However, since using multiple GNSS drastically increases *DPE+*'s computational cost, we limit this algorithm either to GPS and Galileo (Experiment 2) or to GPS only (Experiment 3) and obtain results with the other four algorithms for those scenarios, too, for comparison. Finally, we also apply the algorithms to Watson et al.'s data[5] in Experiment 4 to get insights on the impact of using low-cost hardware in comparison to standard equipment. We only consider GPS for Experiment 4 because the numbers of Galileo and BeiDou satellites were comparatively small in 2018.

For all experiments, we independently process each snapshot. We expect improved positioning accuracy with smoothing [32], but want to be able to transfer our results to scenarios where we capture snapshots at much longer time intervals, e.g., one hour, and the receiver moves up to several kilometres per hour. We initialise each algorithm run with ground truth position and time perturbed by random uniformly distributed errors and a common bias of zero.

[5] https://bit.ly/2vybpgA

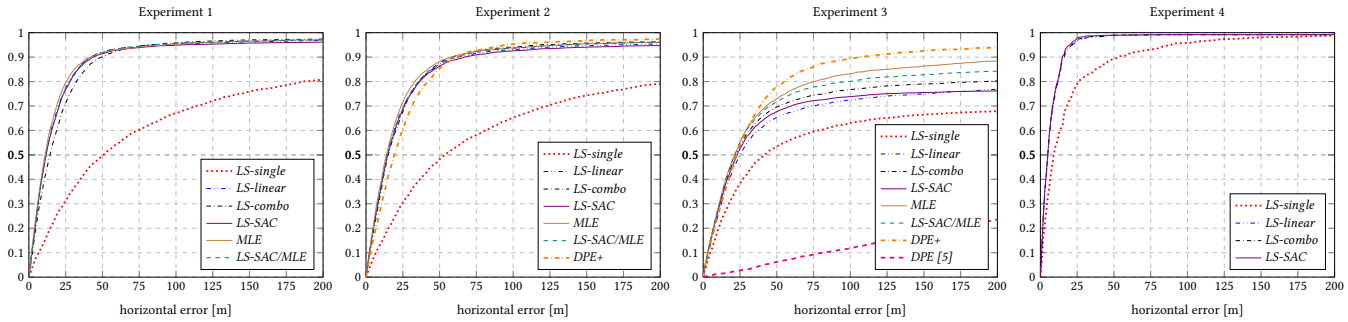

Figure 2: Empirical cumulative distribution functions of the horizontal localisation errors.

Table 1: Selected performance measures.

experiment	data	signals	median horizontal localisation error [m]								portion of horizontal errors < 200 m								mean algorithm runtime [s] per snapshot							
			LS-single	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+*	*DPE-[5]*	*LS-single*	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+*	*DPE-[5]*	*LS-single*	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+*	*DPE-[5]*
1	2	L1, E1, B1C	50.2	11.6	14.4	11.7	11.0	11.7	-	-	81%	97%	97%	96%	97%	97%	-	-	0.51	0.53	0.66	0.52	1.41	0.54	-	-
2	2	L1, E1	53.7	14.4	15.1	14.8	13.9	14.6	19.3	-	79%	95%	96%	95%	96%	96%	97%	-	0.11	0.13	0.24	0.11	0.59	0.13	139.47	-
3	2	L1	42.0	25.0	22.3	23.1	21.4	21.6	22.5	468.0	68%	77%	80%	76%	88%	84%	94%	24%	0.04	0.05	0.21	0.05	0.34	0.09	35.60	842.77
4	5	L1	9.0	5.1	5.1	4.9	-	-	-	-	99%	100%	100%	99%	-	-	-	-	0.02	0.02	0.05	0.02	-	-	-	-

The random initial error is limited to 10 km in latitude and longitude, respectively, vertically to 100 m, and to 1 s in time.

We run Experiment 1, 2 and 3 on a computer with two Intel® Xeon® Gold 5120 CPUs with 2.2 GHz base frequency and Experiment 4 on one with an Intel® Core™ i7-9750H CPU with 2.6 GHz.

5.3 Results

Our main measure to assess algorithm performance is the horizontal positioning error w.r.t. the ground truth tracks[6]. Figure 2 presents this measure as empirical cumulative distribution functions (cdf). For example, a value of 0.6 at the vertical axis for a value of 15 m at the horizontal axis implies that 60% of the positioning errors are less than or equal to 15 m. Since we independently process every snapshot, there are no qualitative differences between the results of the static and dynamic tests, and we combine them in a single diagram for each experiment. An adequate measure for the overall accuracy of an algorithm is the median positioning error, which is the value on the horizontal axis where the cdf crosses 0.5 and also shown in Tab. 1. Furthermore, we choose the portion of errors smaller than 200 m as measure for the reliability and robustness of an algorithm, which is the right-most values of the corresponding cdf in the diagram and listed in Tab. 1.

As third performance measure, we consider the mean algorithm runtime in Tab. 1. The absolute runtime is not meaningful because it significantly depends on factors such as computer hardware and programming language. However, since they are the same for all algorithms, a relative comparison of their runtimes is possible.

[6]The vertical accuracy is less relevant for most applications. Like for conventional GNSS, it is slightly lower than the horizontal one and mainly depends on the quality of the corrections for signal delays in the atmosphere, cf. Sec. 6.

5.4 Discussion

All examined algorithms are comparably reliable in Experiment 4, which offers the best—although, not high—raw signal quality. Satellite acquisition works reliably and no advanced strategy to identify the outliers in the code phases is necessary since the SNR is a good indicator. Just selecting the observations with the highest SNRs as done by *LS-single* leads to reliable and fast positioning even with the non-robust least-squares optimisation. However, our SnapperGPS data collection contains signals from low-cost hardware, which reduces the SNR of the valid code phases. Therefore, algorithms like *LS-combo* and *LS-SAC* that actively search for the set of inliers and depend less on the SNR-based observation ranking are more reliable in Experiments 1–3. *LS-SAC* offers the lower runtime because it maximises the number of satellites that it employs for the final optimisation step, but still executes most operations on a small subset of the acquired satellites. However, the main reason for the first class of our algorithms being a magnitude faster for GPS than methods from the literature is that we tighten the acquisition frequency search space by robustly estimating the device's frequency offset at first. This does not compromise positioning accuracy.

While the first four algorithms perform satellite selection and location estimation in a loosely coupled fashion, *MLE* achieves a similar or better reliability by tightly coupling both tasks. The method is also slightly more accurate because it implicitly weights the valid observations, too. On the other hand, the runtime is larger due to the non-linear optimisation. In practice, *LS-SAC/MLE* is a good choice because it provides position fixes whenever at least one of the underlying algorithms succeeds and the mean runtime is not much higher than for *LS-SAC*.

The unmatched high computation time is also the core disadvantage of *DPE+*, which, on the other hand, achieves the highest

reliability in Experiments 2 and 3. This is due to the superiority of a one-step approach over one with two steps from an information theoretic point of view. The latter discard all information after the acquisition stage except the most likely code phases and the associated SNRs, which they then employ for positioning. In contrast, *DPE+* makes use of all information in the signals for positioning. This allows to incorporate satellites into the position estimation that show only a local but not a global peak at the correct code phase in the acquisition correlogram. Note that the adjustments made in Sec. 4.4 to Bissig et al.'s algorithm *DPE-[5]* are crucial. The signal quality is too low to reliably estimate locations for individual one-millisecond chunks of a twelve-millisecond snapshot. So, averaging twelve solutions usually does not lead to a correct fix.

In the few percent of the recordings where none of the two-step approaches finds a plausible solution, the preceding satellite acquisition stage just does not provide enough satellites with correct code phases to solve the five-dimensional optimisation problem. The existence of such failures is expected because the data contains scenarios with limited or no sky visibility due to tree coverage, narrow roads, and especially underpasses and bridges. While *DPE+* does not have an acquisition phase, it still fails when not enough satellites are in view to establish a unique peak in its search space.

Employing multiple GNSS simply increases the number of usable satellites in view, thus significantly reducing the count of failed fixes in Experiment 1 and 2 in contrast to Experiment 3. This underpins the importance of using multiple GNSS for positioning under challenging conditions. More satellites improve accuracy, too, because more observations help to average out imprecisions.

6 SNAPPERGPS WEB APPLICATION

The previously introduced and evaluated algorithms are the backbone of the SnapperGPS website[3], which exposes three core interactive views to the user, cf. Fig. 3:

(1) a *configuration view* to connect a SnapperGPS receiver via WebUSB and configure it for a subsequent deployment,
(2) an *upload view* to transfer recorded snapshots and timestamps from a connected device to a server, and
(3) a *download view* that shows the track on an interactive map and offers downloads in various file formats after the server calculated the positions for a set of snapshots.

The core tasks of the back-end are:

(1) to pull satellite navigation data, i.e., ephemerides and almanacs, from a public NASA database and
(2) to calculate locations for uploaded snapshot sets.

Task (2) is done by a custom Python back-end[1]. The set of raw snapshots, the corresponding timestamps, the navigation data for these days, and a user-provided start location of the track are passed to the back-end, which then computes a location estimate, an associated uncertainty, and a corrected timestamp for each snapshot using algorithms from Sec. 4. In addition to GNSS snapshots, SnapperGPS optionally takes into account other types of observations:

- It can employ pressure and temperature measurements from on-board sensors and the hypsometric formula to calculate altitudes that it then treats as additional range observations for positioning, potentially with a different weight/variance than the actual satellite pseudorange observations.
- It can feed the same pressure and temperature measurements into one of three models that correct pseudorange predictions for signal propagation delays in the troposphere [12, 19, 27], cf. Eq. (1).
- It can use ionosphere parameters from GPS almanacs in one of two models of the delay in the ionosphere [13, 27].
- It can derive another altitude observation given a latitude and longitude by combining a geoid model of the Earth [15, 20] with a digital elevation model [2, 8].

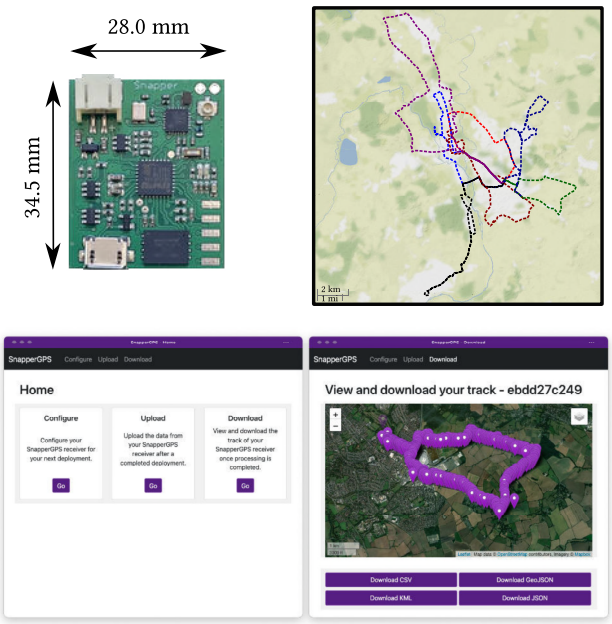

Figure 3: A SnapperGPS board, ground truth tracks of our dynamic tests, and exemplary views of the web application.

7 CONCLUSIONS

This work presented tailored algorithms that enable reliable location estimation from short GNSS signal snapshots with ultra-low resolution. They allow us and others to build energy-efficient GNSS receivers at unmatched low costs, which is ideal for applications like wildlife, fitness, or certain logistics tracking. SnapperGPS is an easy-to-use system that comprises an open implementation of the algorithms and a manufacturing-ready receiver design and was tested on data collected in a broad range of real-world scenarios.

We look forward to evaluate SnapperGPS in wildlife tracking field trials over several months jointly with researchers from conservation science, zoology, and ecology. We hope to receive feedback on the usefulness, reliability, and accessibility of our system and plan to improve it accordingly. Once the feedback is positive, we will set up an open scheme that allows a broader audience to order affordable SnapperGPS receivers for their own research without having to care about the manufacturing themselves. We also plan

to introduce and evaluate additional features such as snapshot capturing triggered by an activity sensor or a salt-water switch and the combination with a sensor that records data that benefits from GNSS-timestamping with millisecond accuracy, such as audio or acceleration data that shall be synchronised.

ACKNOWLEDGMENTS

SnapperGPS is a joined project by Amanda Matthes, Jonas Beuchert, and Alex Rogers and supported by an EPSRC IAA Technology Fund. Additionally, the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems supports Amanda and Jonas.

REFERENCES

[1] David Akopian. 2005. Fast FFT based GPS satellite acquisition methods. *IEE Proceedings–Radar, Sonar and Navigation* 152, 4 (2005), 277–286.
[2] Christopher Amante and Barry W. Eakins. 2009. *ETOPO1 arc-minute global relief model: procedures, data sources and analysis.* Vol. NESDIS NGDC-24. National Geophysical Data Center, Boulder, CO, USA.
[3] Penina Axelrad, Ben K. Bradley, James Donna, Megan Mitchell, and Shan Mohiuddin. 2011. Collective Detection and Direct Positioning Using Multiple GNSS Satellites. *Navigation* 58, 4 (2011), 305–321.
[4] Baseband Technologies Inc. [n.d.]. *Snapshot Positioning - Next Generation GNSS Receiver for Low Power Applications.* Baseband Technologies Inc. Retrieved March 24, 2020 from http://www.basebandtech.com/wp-content/uploads/Snapshot-Receiver.pdf
[5] Pascal Bissig, Manuel Eichelberger, and Roger Wattenhofer. 2017. Fast and robust GPS fix using one millisecond of data. In *16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN).* ACM Press, New York, NY, USA, 223–234.
[6] Kai Borre, Dennis M. Akos, Nicolaj Bertelsen, Peter Rinder, and Søren H. Jensen. 2007. *A software-defined GPS and Galileo receiver: a single-frequency approach.* Springer Science & Business Media, New York, NY, USA.
[7] Pau Closas Gómez. 2009. *Bayesian signal processing techniques for GNSS receivers: from multipath mitigation to positioning.* Ph.D. Dissertation. Universitat Politècnica de Catalunya.
[8] Earth Resources Observation And Science (EROS) Center. 2017. Shuttle Radar Topography Mission (SRTM) 1 Arc-Second Global. https://www.usgs.gov/centers/eros/science/usgs-eros-archive-digital-elevation-shuttle-radar-topography-mission-srtm-1-arc?qt-science_center_objects=0#qt-science_center_objects
[9] Manuel Eichelberger, Ferdinand von Hagen, and Roger Wattenhofer. 2019. Multi-Year GPS Tracking Using a Coin Cell. In *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications* (Santa Cruz, CA, USA) *(HotMobile '19).* ACM, New York, NY, USA, 141–146.
[10] Ignacio Fernández-Hernández and Kai Borre. 2016. Snapshot positioning without initial information. *GPS Solutions* 20, 4 (March 2016), 605–616.
[11] Martin A. Fischler and Robert C. Bolles. 1981. Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Commun. ACM* 24, 6 (1981), 381–395.
[12] Clyde C. Goad and L. Goodman. 1974. A Modified Tropospheric Refraction Correction Model. In *American Geophysical Union Annual Fall Meeting.* San Francisco, CA, USA.
[13] John A Klobuchar. 1996. Ionospheric effects on GPS. *Global Positioning System: theory and application* 136 (1996), 513–514.
[14] Philip Kwan. 2019. *NAVSTAR GPS Space Segment/Navigation User Segment Interfaces (IS-GPS-200K).* National Coordination Office for Space-Based Positioning, Navigation, and Timing. https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf
[15] Frank G. Lemoine, Steve C. Kenyon, John K. Factor, Ronald G. Trimmer, Nikolaos K. Pavlis, Douglas S. Chinn, Christopher M. Cox, Steven M. Klosko, Scott B. Luthcke, Mark H. Torrence, et al. 1998. *The development of the joint NASA GSFC and NIMA geopotential model EGM96.* Technical Report. NASA Goddard Space Flight Center, Greenbelt, MD, USA.
[16] Jie Liu, Bodhi Priyantha, Ted Hart, Heitor S. Ramos, Antonio A. F. Loureiro, and Qiang Wang. 2012. Energy efficient GPS sensing with cloud offloading. In *Proceedings of the 10th ACM Conference on Embedded Network Sensor Systems (SenSys)* (Toronto, Ontario, Canada). ACM Press, New York, NY, USA, 85–98.
[17] Laura A. McMahon, Janet L. Rachlow, Lisa A. Shipley, Jennifer S. Forbey, Timothy R. Johnson, and Peter J. Olsoy. 2017. Evaluation of micro-GPS receivers for tracking small-bodied mammals. *PLOS ONE* 12, 3 (March 2017).
[18] Craig Morrison. [n.d.]. *ATS G10 Ultralite GPS.* Advanced Telemetry Systems Australia. Retrieved March 24, 2020 from https://nebula.wsimg.com/2c21477b3a3e90c8279c4b5979312195?AccessKeyId=CEA7BFCD16F82D670903&disposition=0
[19] Bradford W. Parkinson, Per Enge, Penina Axelrad, and James J. Spilker Jr. 1996. *Global positioning system: Theory and applications, Volume II.* American Institute of Aeronautics and Astronautics, Washington, DC, USA.
[20] Nikolaos K. Pavlis, Simon A. Holmes, Steve C. Kenyon, and John K. Factor. 2012. The development and evaluation of the Earth Gravitational Model 2008 (EGM2008). *Journal of geophysical research: solid earth* 117, B4 (2012).
[21] RTCA Inc. 2006. *Minimum Operational Performance Standards for Global Positioning System/Wide Area Augmentation System Airborne Equipment.* RTCA Inc.
[22] Jaume Sanz Subirana, José M. Juan Zornoza, and Mamuel Hernández-Pajares. 2011. *Best Linear Unbiased Minimum-Variance Estimator (BLUE).* European Space Agency. https://gssc.esa.int/navipedia/index.php/Best_Linear_Unbiased_Minimum-Variance_Estimator_(BLUE)
[23] Kelly C. Seals. 2014. *Enhanced Acquisition Techniques for GPS L1C Receivers.* Ph.D. Dissertation. Worcester Polytechnic Institute.
[24] Bashir A. Siddiqui, Jie Zhang, Mohammad Z. H. Bhuiyan, and Elena S. Lohan. 2010. Joint data-pilot acquisition and tracking of Galileo E1 open service signal. In *Ubiquitous Positioning Indoor Navigation and Location Based Service.* IEEE, Finland, 1–7.
[25] Andrew Simsky and Frank Boon. 2003. Carrier phase and Doppler-based algorithms for real-time standalone positioning. In *Proceedings of GNSS.* Austrian Institute of Navigation, Graz, Austria, 1–25.
[26] James J. Spilker Jr. 1977. *Digital communications by satellite.* Prentice Hall, Englewood Cliffs, NJ, USA.
[27] James B.-Y. Tsui. 2004. *Fundamentals of Global Positioning System Receivers.* John Wiley & Sons, Inc., Newy York, NY, USA.
[28] Keith Van Dierendonck, Ming Xu, and Pejman L. Kazemi. 2015. System, method, and computer program for a low power and low cost GNSS receiver. US Patent 9,116,234.
[29] Frank van Diggelen. 2009. *A-GPS: Assisted GPS, GNSS, and SBAS.* Artech House, Norwood, MA, USA.
[30] Ryan M. Watson, Jason N. Gross, Clark N. Taylor, and Robert C. Leishman. 2019. Enabling robust state estimation through measurement error covariance adaptation. *IEEE Trans. Aerospace Electron. Systems* 56, 3 (2019), 2026–2040.
[31] Shulin Yan, Shenghong Zhou, and Yuguo Zhang. 2018. An efficient two-stage B1C signal acquisition technique for engineering implementation of the modern beidou receiver. In *4th International Conference on Computer and Technology Applications (ICCTA).* IEEE, Istanbul, Turkey, 46–53.
[32] Emanuele Ziglioli and Eugenio Realini. 2015. Extending goGPS for Snapshot Positioning. *Geomatics Workbooks* 12 (2015), 383–393.

A SATELLITE RELIABILITY UPDATE FOR *LS-SAC*

Algorithm 2 (*LS-SAC*) operates on subsets $S \in \mathcal{P}(\{1, 2, \dots N\})$ of the full set of potentially visible satellites with indices $1, 2, \dots N$. In line 1, it defines the size $|S| = M$ of the subsets. At the start of each iteration in line 5, the algorithm calculates the probability for each untested subset of this size to contain only inliers. In the first iteration, this is solely based on the SNR-induced priors

$$P(V_1 = 1 \mid \mathrm{SNR}_1, \dots \mathrm{SNR}_N) = \prod_{i \in S} P(v_i = 1 \mid \mathrm{SNR}_i), \tag{4}$$

where $V_1 = 1 \Leftrightarrow v_i = 1 \, \forall \, i \in S$ indicates the event that all code phases of the satellites in S are inliers. The algorithm selects the most promising subset

$$S_1 = \underset{S \in \mathcal{P}(\{1,\dots N\}), |S|=M}{\operatorname{argmax}} P(V_1 = 1 \mid \mathrm{SNR}_1, \dots \mathrm{SNR}_N) \tag{5}$$

to test next. A second iteration of the loop (lines 3–20) is only carried out if the first iteration with subset S_1 does not produce a plausible solution, i.e., $V_1 = 0$ if we assume that the least-squares optimisation produces a plausible solution if and only if no outlier exists in S_1. The failed first iteration provides additional information for the calculation of the probabilities for each subset in line 5 of

the second iteration, i.e.,

$$P\left(V_2 = 1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1 = 0\right) = \prod_{i \in S} P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0\right). \tag{6}$$

In a general iteration k, this becomes

$$P\left(V_k = 1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1 = 0, \ldots V_{k-1} = 0\right) = \prod_{i \in S} P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right). \tag{7}$$

For an efficient computation of $P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right)$ in every iteration k, we want to derive it by updating the corresponding term $P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)$ from the previous iteration $k-1$. The equation for the latter conditioned on the outcome of the $(k-1)$-th plausibility test is

$$\begin{aligned}
&P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \\
&= P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 0\right) \\
&\quad \cdot P\left(V_{k-1} = 0 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \\
&\quad + P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\
&\quad \cdot P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \\
&= P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right) \\
&\quad \cdot \left(1 - P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right) \\
&\quad + P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\
&\quad \cdot P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)
\end{aligned} \tag{8}$$

$$\begin{aligned}
\Leftrightarrow\ &P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 1\right) \\
&= \left(\left(P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right.\right. \\
&\quad - P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\
&\quad \left.\left. \cdot\ P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right)\right. \\
&\quad \left./\left(1 - P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right).\right.
\end{aligned} \tag{9}$$

The only term in this equation that we cannot immediately take from the previous iteration $k-1$ is

$$\begin{aligned}
&P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\
&= \begin{cases} 1, & \text{if } s \in S_{k-1}, \\ P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1, s \notin S_{k-1}\right), & \\ \quad \text{otherwise}, & \end{cases} \\
&\approx \begin{cases} 1, & \text{if } s \in S_{k-1}, \\ P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right), & \text{otherwise}, \end{cases}
\end{aligned} \tag{10}$$

where we approximate the second case such that substitution in Eq. (9) gives

$$\begin{aligned}
&P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right) \\
&\approx \begin{cases} \frac{P\left(v_i=1 \mid \mathrm{SNR}_i, V_1=0, \ldots V_{k-2}=0\right) - P\left(V_{k-1}=1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1=0, \ldots V_{k-2}=0\right)}{1 - P\left(V_{k-1}=1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1=0, \ldots V_{k-2}=0\right)}, & \\ \quad \text{if } i \in S_{k-1}, & \\ P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right), & \text{otherwise}, \end{cases}
\end{aligned} \tag{11}$$

which is the desired update equation. With Eq. (11) and Eq. (7), we can recursively recalculate the probability to contain only inliers for each subset given all available information in line 5 of Alg. 2.

B ALGORITHM PARAMETRISATIONS

Table 2: Algorithm parameters, the algorithms for which they are valid, and their values. We tuned the latter to achieve good performance on data from a few stationary SnapperGPS receiver tests. This data is not part of the data collections for the evaluation in Sec. 5 to avoid reporting too optimistic performance results due to overfitting.

parameter	*LS-single*	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+ & DPE-[5]*	value
elevation mask	x	x	x	x	x	x	x	10°
points for code-phase interpolation	x	x	x	x	x	x		3
max. number of satellites	x							5
max. number of satellites		x		x		x		15
max. number of satellites			x					10
temporal search space					x	x	x	±1 s
horizontal search space					x	x	x	[±10 km, ±10 km]
vertical search space					x	x	x	±100 m
temporal search space resolution							x	±40 ms
least-squares iterations	x	x	x	x		x		3
number of points for averaging							x	$2^4 = 16$
time-out							x	30 min
max. tolerated residuals	x	x	x	x		x		200 m
max. tolerated horz. movement	x	x	x	x		x		15 km
max. tolerated clock change	x	x	x	x		x		30 s
pseudorange uncertainty						x		20 m
max. tolerated uncertainty						x		200 m
code-phase uncertainty					x	x		26.67, 6.67, 1.67, 0.42, 0.10, 0.03 µs
max. number of RANSAC iterations				x		x		3