Probabilistic Snapshot GNSS for Low-Cost Wildlife Tracking

Jonas Beuchert

Kellogg College

University of Oxford

A thesis submitted for the degree of

Doctor of Philosophy

Trinity 2023

Acknowledgements

Jonas Beuchert was funded by the EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines and Systems (University of Oxford project code: DFT00350-DF03.01, UKRI/EPSRC grant reference: EP/S024050/1).

The implementation and evaluation of SnapperGPS were co-funded by two EPSRC IAA Technology Funds (University of Oxford project codes: D4D00010-BL14 and D4D00190-BL03.01, UKRI/EPSRC grant references: unknown and EP/X525777/1) and a Kellogg College research grant.

The article processing charges for the publication in the Journal of Open Hardware were covered by an open access UKRI block grant via the University of Oxford.

The presentation during the IEEE International Conference on Robotics and Automation 2023 was co-funded by a Kellogg College travel grant.

The SnapperGPS project was supervised by Alex Rogers from the University of Oxford. He deserves credit for the initial idea of SnapperGPS and his contributions to the development of the initial hardware and firmware need to be emphasised, in particular.

The initial development of the SnapperGPS web application was supported by Peter Prince from Open Acoustic Devices in spring 2021.

The SnapperGPS deployment on loggerhead sea turtles in summer 2021 was conducted in cooperation with Amanda Matthes from the University of Oxford, Alasdair Davies from the Arribada Initiative, and Juan Patino-Martinez and many others from the Maio Biodiversity Foundation.

The SnapperGPS deployment on green sea turtles in spring 2023 was conducted in cooperation with Nguyen Lam Thuy Ngân and Meredith Palmer from Fauna & Flora International with support from the Núi Chúa National Park and local volunteers.

Many people around the world contributed to the SnapperGPS project by trialling the system and providing feedback. The distribution of the hardware to those users was jointly organised with Amanda Matthes, who also designed enclosures for early SnapperGPS receiver versions and contributed to joint write-ups.

The robotics project (Chapter 8) was supervised by Marco Camurri from the University of Oxford (until 2022) / the Free University of Bozen-Bolzano (since 2023) and Maurice Fallon from the University of Oxford. The project idea was

developed jointly with both and the final real-time implementation was completed together with Marco, based on a visual-inertial-lidar state-estimation framework by David Wisth from the University of Oxford, Marco, and Maurice. Marco and other members of the Dynamic Robot Systems Group in the Oxford Robotics Institute supported related field trials.

Several individuals provided feedback on paper drafts and/or chapters of this thesis, including Pavlo (Paul) Gaiduk, Shu Ishida, Matías Mattamala, Marco Camurri, Maurice Fallon, and Alex Rogers.

Abstract

Snapshot GNSS is more energy-efficient than conventional localisation methods based on global navigation satellite systems (GNSS), like the GPS. This is beneficial for long deployments on battery such as in wildlife tracking. However, only a few snapshot GNSS systems that could be used for wildlife tracking have been presented and all have disadvantages. Most significantly, they are closed-source and either not available or expensive. A reason is that they typically require GNSS signals to be captured with good resolution, which demands complex receiver hardware capable of capturing multi-bit data at sampling rates of 16 MHz and more. By contrast, this thesis presents fast algorithms that reliably estimate locations from twelve-millisecond signals that are sampled at just 4 MHz and quantised with only a single bit. This allows to build a snapshot receiver at an unmatched low cost of less than $30 and with particularly low power consumption, outperforming existing systems and enabling low-budget and long-term field work. The system can acquire two positions per hour for a year on a tiny 40 mA h battery. On a challenging public dataset with thousands of snapshots from real-world scenarios, median accuracy is 11 m, comparable to more complex and expensive solutions with higher energy consumption. Additionally, the system has been deployed for several wildlife tracking studies, including on sea turtles, where brief signal acquisition times are crucial to obtain location fixes during surfacing events lasting only 1–2 s. For the first time, (i) snapshot GNSS receiver hardware and (ii) an accompanying cloud-based processing platform are open-source. This allowed several third parties to independently replicate the system. In total, several hundred receivers have been built and millions of locations estimated for those. As three additional contributions, this thesis presents (i) the first evaluation of snapshot GNSS for wildlife tracking across a variety of species and habitats, (ii) the first snapshot GNSS system with cloud-offloading via a low-power narrow-band cellular connection, and (iii) a demonstration of the potential of smoothing for snapshot GNSS. A final contribution are factor graph optimisation algorithms to (i) smooth snapshot GNSS data and (ii) tightly fuse raw GNSS data with inertial measurements and, optionally, lidar observations for precise and smooth localisation. In several environments with little sky visibility, such as a forest, the accuracy of the fused location estimates in the global Earth frame is still 1–2 m, while the estimated trajectories are discontinuity-free and smooth. This requires a professional-grade (non-snapshot) GNSS receiver, but, unlike traditional differential GNSS, no connection to a base station.

Contents

List of Abbreviations **ix**

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Contributions . . . 4
1.2.1 Robust Snapshot GNSS Algorithms for Low-Cost Hardware 4
1.2.2 Low-Cost Open-Source Snapshot GNSS Receiver Hardware . 5
1.2.3 Evaluation of Snapshot GNSS for Various Wildlife Tracking Scenarios . . . 5
1.2.4 A Snapshot GNSS System with Cloud-Offloading via a Narrow-Band Cellular Connection . . . 5
1.2.5 Evaluation of Smoothing of Snapshot GNSS Data . . . 6
1.2.6 Factor Graph Algorithms for Fusion of Raw GNSS Data with Other Sensing Modalities . . . 6
1.3 Thesis Overview . . . 11

2 Background **13**
2.1 Satellite Navigation . . . 14
2.2 GNSS Signals . . . 15
2.3 GNSS Receiver Front-Ends . . . 16
2.4 Satellite Acquisition . . . 18
2.5 Coarse-Time Navigation . . . 22
2.6 Direct Positioning . . . 24
2.7 Snapshot GNSS Systems . . . 26
2.8 GNSS Observables . . . 28
2.9 Code-Based Positioning . . . 31
2.10 Assisted GNSS . . . 34
2.11 Carrier-Based Positioning . . . 34
2.12 Sensor Fusion and Factor Graph Optimisation . . . 36
2.13 Factor Graph Optimisation for Satellite Navigation . . . 38
2.14 Satellite Navigation for Wildlife Tracking . . . 40
2.15 Tracking of Aquatic Animals . . . 42

3 Algorithms for Energy-Efficient Low-Cost Location Estimation Using GNSS Signal Snapshots 46
3.1 Problem Statement . . . 46
3.2 Robust Algorithms . . . 48
3.2.1 Snapshot-Based Satellite Acquisition . . . 50
3.2.2 Satellite Selection for Coarse-Time Navigation . . . 54
3.2.3 Code-Based Positioning via Maximum-Likelihood Estimation 59
3.2.4 Direct Positioning . . . 62
3.3 Evaluation . . . 63
3.3.1 Data . . . 63
3.3.2 Experiments . . . 64
3.3.3 Results . . . 65
3.3.4 Discussion . . . 67
3.4 Web Application . . . 70
3.5 Conclusions . . . 73

4 Open Hardware for Energy-Efficient Low-Cost Wildlife Location Tracking with Snapshot GNSS 74
4.1 Problem Statement . . . 74
4.2 Electronics . . . 78
4.2.1 Antenna . . . 78
4.2.2 Radio . . . 80
4.2.3 Microcontroller . . . 81
4.2.4 External Flash Memory . . . 81
4.2.5 Interfaces . . . 81
4.2.6 Power Supply . . . 83
4.2.7 Printed Circuit Board . . . 85
4.2.8 Bill of Materials . . . 86
4.3 Firmware . . . 91
4.4 Web Application . . . 96
4.4.1 Interface between Web Application and Receiver . . . 96
4.4.2 Automatic Calibration . . . 97
4.5 Evaluation . . . 98
4.5.1 Accuracy . . . 98
4.5.2 Energy Consumption . . . 99
4.6 Conclusions . . . 101

5 Deployments of Low-Cost Snapshot GNSS Receivers for Wildlife Tracking 103
5.1 Problem Statement . . . 103
5.2 Selected Deployments . . . 104
5.2.1 Tracking of a Large Population of Sea Turtles . . . 104
5.2.2 Tracking of Sea Turtles on a Small Budget . . . 108
5.2.3 Long-Term Tracking of Terrestrial and Freshwater Turtles . 109
5.2.4 Non-Obtrusive Tracking of Birds . . . 111
5.2.5 Cost-Efficient Tracking of Goats . . . 111
5.2.6 Research and Development Platform . . . 112
5.3 User Feedback . . . 112
5.3.1 Overall Interest . . . 113
5.3.2 Longer Deployments on Small Animals . . . 115
5.3.3 Studies with Larger Sample Size on the Same Budget . . . 117
5.3.4 Uptake of Wildlife Tracking by Conservationists and Researchers for Whom Costs were Previously Too High . . . 118
5.3.5 Tracking of Aquatic Animals that Surface Only Briefly . . . 121
5.3.6 Customisation of Tags to Cater Specific Deployment Needs . 122
5.3.7 Problems and Suggestions for Future Work . . . 124
5.4 Conclusions . . . 127

6 A Low-Cost Snapshot GNSS Receiver with Cloud-Offloading via a Narrow-Band Cellular Connection 129
6.1 Problem Statement . . . 129
6.2 System Design . . . 132
6.2.1 Wireless Technology . . . 132
6.2.2 Electronics . . . 134
6.2.3 Software . . . 136
6.3 Evaluation . . . 137
6.3.1 Experiments . . . 137
6.3.2 Results . . . 138
6.3.3 Discussion . . . 140
6.4 Conclusions . . . 142

7 Smoothing Algorithms for High-Rate Snapshot GNSS 143
7.1 Problem Statement . . . 143
7.2 Models and Algorithms . . . 144
7.2.1 Rauch-Tung-Striebel Smoother . . . 145
7.2.2 Gaussian Process Regression . . . 147
7.2.3 Tightly Coupled Factor Graph Optimisation . . . 148

7.3 Evaluation . . . 150
7.3.1 Data . . . 151
7.3.2 Experiments . . . 152
7.3.3 Results . . . 152
7.3.4 Discussion . . . 154
7.4 Conclusions . . . 156

8 Factor Graph Fusion of Raw GNSS Sensing with IMU and Lidar for Precise Mobile Localisation without a Base Station 159
8.1 Problem Statement . . . 159
8.2 System Description . . . 161
8.2.1 Frames . . . 162
8.2.2 State . . . 163
8.2.3 Measurements . . . 163
8.2.4 Maximum-a-Posteriori Estimation . . . 164
8.3 Factor Graph Formulation . . . 164
8.3.1 Pre-Integrated Inertial Measurements . . . 165
8.3.2 ICP Registration . . . 165
8.3.3 Pseudoranges . . . 165
8.3.4 Carrier Phases . . . 166
8.4 Implementation . . . 168
8.5 Evaluation . . . 168
8.5.1 Experiment 1: Raw GNSS and IMU fusion . . . 171
8.5.2 Experiment 2: Raw GNSS, IMU, and Lidar Fusion . . . 172
8.5.3 Experiment 3: Carrier-Phase-Only Fusion . . . 173
8.6 Conclusions . . . 173

9 Conclusions 174
9.1 Summary . . . 174
9.2 Outlook . . . 176

References 179

List of Abbreviations

GNSS	Global navigation satellite system
GPS	Global Positioning System (USA)
GLONASS	GLObal NAvigation Satellite System (Russia)
BDS	BeiDou Navigation Satellite System (China)
OSHWA	Open Source Hardware Association
lidar	Light detection and ranging / laser imaging, detection, and ranging
IMU	Inertial measurement unit
PCB	Printed circuit board
CTN	Coarse-time navigation
A-GNSS	Assisted GNSS
C/A code	Coarse/acquisition code
DC	Direct current
AC	Alternating current
IF	Intermediate frequency
I component/signal	In-phase component/signal
Q component/signal	Quadrature component/signal
IQ signal	In-phase and quadrature signal
ADC	Analogue-to-digital converter
SDR	Software-defined receiver/radio
PCPS	Parallel code phase search
RF	Radio frequency
FFT	Fast Fourier transformation
SNR	Signal-to-noise ratio
DPE	Direct position estimation

DOP . . . . . Dilution of precision
HDOP . . . . . Horizontal dilution of precision
CD . . . . . Collective detection
GPU . . . . . Graphics processing unit
CPU . . . . . Central processing unit
PARC . . . . . Parallel input capture interface
DMA . . . . . Direct memory access
SPS . . . . . Standard positioning service
DGNSS . . . . . Differential GNSS
RTK . . . . . Real-time kinematics
PPP . . . . . Precise point positioning
NTRIP . . . . . Networked Transport of RTCM via Internet Protocol
RTCM . . . . . Radio Technical Commission for Maritime Services
LAMBDA . . . . . Least-squares ambiguity decorrelation adjustment
MAP . . . . . Maximum a-posteriori
EKF . . . . . Extended Kalman filter
FGO . . . . . Factor graph optimisation
UAV . . . . . Unmanned aerial vehicle
MSCKF . . . . . Multi-state constraint Kalman filter
RMSE . . . . . Root-mean-square error
VHF . . . . . Very high frequency (30–300 MHz)
LS . . . . . Least-squares
MLE . . . . . Maximum likelihood estimation
RANSAC . . . . . Random sample consensus
SAC . . . . . Sample consenus
BFGS algorithm . Broyden–Fletcher–Goldfarb–Shanno algorithm
ENU coordinates . East-north-up coordinates
CDF . . . . . Cumulative distribution function
RAM . . . . . Random-access memory
MCU . . . . . Microcontroller unit
LED . . . . . Light-emitting diode

USB . . . . . Universal Serial Bus

GPIO . . . . . General-purpose input/output pin

IC . . . . . Integrated circuit

LNA . . . . . Low-noise amplifier

LiPo battery . . . . Lithium-ion polymer battery

VCO . . . . . Voltage-controlled oscillator

TCXO . . . . . Temperature-compensated crystal oscillator

SAW filter . . . . . Surface acoustic wave filter

SPI . . . . . SCSI Parallel Interface

RC oscillator . . . . Resistor-capacitor oscillator

JST . . . . . Japan Solderless Terminal

MOSFET . . . . . Metal-oxide-semiconductor field-effect transistor

N-MOSFET . . . . N-channel MOSFET

P-MOSFET . . . . P-channel MOSFET

GND . . . . . Ground

BOM . . . . . Bill of materials

LDC . . . . . Inductance-to-digital converter

CNC . . . . . Computer numerical control

POM . . . . . Polyoxymethylene

I2C . . . . . Inter-Integrated Circuit

SWD . . . . . Serial Wire Debug

EM . . . . . Energy mode

RTC . . . . . Real-time counter

USART . . . . . Universal synchronous and asynchronous receiver-transmitter

CRC . . . . . Cyclic redundancy check

SRAM . . . . . Static random-access memory

PWA . . . . . Progressive web app

LPWAN . . . . . Low-power wide-are network

DSP . . . . . Digital signal processor

FPGA . . . . . Field-programmable gate array

GDP . . . . . Gross domestic product

EDA . . . . . Electronic design automation

LTE . . . . . Long-Term Evolution (wireless broadband standard)

LTE-M . . . . . Long-Term Evolution Machine Type Communication (LPWAN standard)

LoRaWAN . . . . . Long Range Wide Area Network

IoT . . . . . Internet of things

eMTC . . . . . Enhanced Machine Type Communication (LPWAN standard)

NB-IoT . . . . . Narrowband internet of things (LPWAN standard)

JSON . . . . . JavaScript Object Notation

HMM . . . . . Hidden Markov model

RTS smoother . . . Rauch–Tung–Striebel smoother

MCMC . . . . . Markov chain Monte Carlo

GP . . . . . Gaussian process

GPR . . . . . Gaussian process regression

ECEF . . . . . Earth-centred-Earth-fixed

EKF . . . . . Extended Kalman filter

ICP . . . . . Iterative closest point

1 Introduction

1.1 Motivation

Global navigation satellite systems (GNSS), e.g., the Global Positioning System (GPS), enable localisation of objects, humans, and animals anywhere on the surface of the Earth. For example, biologists, ecologists, and conservationists tag wild animals with GNSS receivers to track their movements and gain valuable insights into their behaviour and their responses to external forces such as climate change and human activity [1–13]. However, such deployments of GNSS tags are currently limited by several factors:

1. *Size and weight.* Conventional GNSS tags consume a substantial amount of energy for a fix, requiring bulky batteries for all but short deployments. For animal welfare reasons, this prohibits tagging small animals for extended periods [1, 6–8, 14, 15].

2. *Operation time.* The substantial energy consumption also prevents very long deployments on all but large species [1, 7, 14, 16]. For deployments lasting months or years, researchers often resort to non-GNSS tags like light-level geolocators with magnitudes worse tracking accuracy [6].

3. *Cost.* Commercial wildlife tracking tags range from a few hundred to more than ten thousand USD [2, 6, 15]. Since drawing conclusions on a population level usually requires a minimum sample size of 30 or more, these costs jeopardise the statistically significance of findings [2, 6, 17, 18]. In addition, they exclude organisations and individuals with a low budget from using the technology [2].

4. *Time to fix.* A conventional duty-cycled GNSS tag usually requires several seconds for a fix under good conditions, more with limited sky visibility, and 30 s or more once every few hours [19]. This is too long for certain deployments, e.g., on an aquatic animal, where surfacing events lasting just 1–2 s are the only opportunities to collect GNSS data.

5. *Customisation potential.* Answering novel research questions frequently requires a unique combination of sensing modalities or a deployment setting that has not been considered previously. However, off-the-shelf GNSS tags commonly do not allow for end-user customisation [2].

A core reason behind several of these points is that conventional GNSS receivers decode satellite signals and compute locations in real time on the device itself. Hence, their signal acquisition and processing can take a long time and is computational and energy intensive. If operated on a battery, their lifetime is limited and/or they are bulky [1, 6–8]. Beyond wildlife tracking, similar issues are evident in other application domains such as wearable fitness and low-cost logistics tracking.

So-called snapshot receivers address these challenges [1, 6, 14, 20]. They 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 can 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, cf. Figure 1.1. This eliminates almost any on-board signal processing, greatly lowering hardware complexity and energy consumption. However, the processing

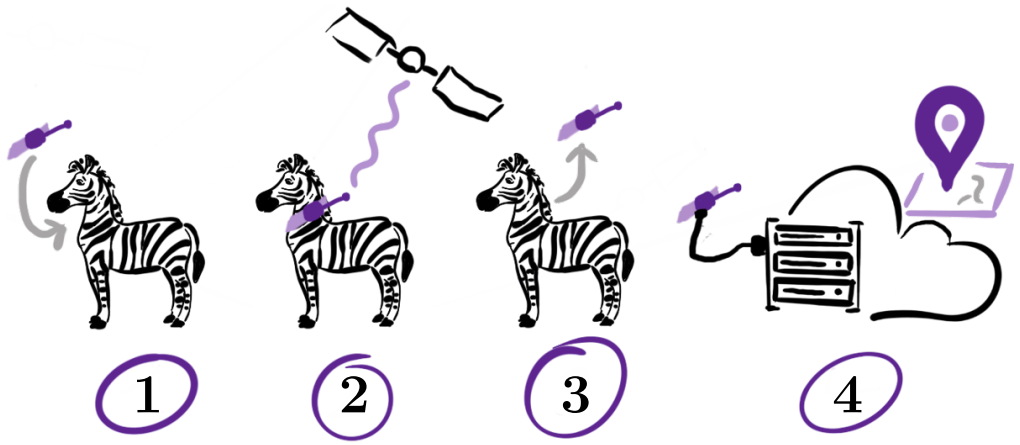


Figure 1.1: Snapshot GNSS workflow: 1) Deploy receiver. 2) Receiver captures GNSS signal snapshots. 3) Recover receiver. 4) Cloud-based algorithm estimates position history.

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. Solving this challenge requires specialised algorithms.

Despite their potential advantages, snapshot receivers are currently not widely used for wildlife tracking and no system is available to practitioners in the field at low or medium costs (<$3000) [6]. Therefore, the core of this thesis covers the development of a low-power, low-cost, open-source snapshot GNSS receiver for wildlife tracking to benefit:

1. Longer deployments on small animals that cannot carry bulky batteries,
2. Studies with larger sample size on the same budget,
3. Uptake of wildlife tracking by conservationists and researchers for whom costs were previously too high, especially, in the Global South,
4. Tracking of aquatic animals that surface only briefly, and
5. Customisation of tags to cater specific deployment needs.

However, the contributions of this thesis have relevancy to a more general class of problems beyond wildlife tracking. In most systems that incorporate satellite navigation for localisation—not just in wildlife tracking tags—a single GNSS module performs all signal processing steps from signal acquisition to location estimation

(often with additional filtering assuming a particular movement model and use case). System designers often treat the module as a black box that outputs position fixes. Depending on the application, these fixes are then logged, transmitted to a user, or fused with measurements of other sensors that are part of the same system. In contrast, this dissertation describes localisation systems that leverage raw GNSS receiver data instead of pre-computed fixes. In a separate module, they perform localisation by applying probabilistic signal processing and optimisation to this raw data. This results in systems that are either more energy efficient or provide more robust localisation than comparable systems that integrate GNSS receiver modules in the aforementioned black-box fashion.

1.2 Contributions

The contributions of this thesis range from hardware designs to the introduction of new algorithms and are outlined in the following.

1.2.1 Robust Snapshot GNSS Algorithms for Low-Cost Hardware

The first project covered in the dissertation is SnapperGPS, a novel snapshot GNSS system targeted at wildlife tracking. As part of this, the thesis introduces fast algorithms that reliably estimate positions from short signal snapshots that were captured with a much lower sampling frequency and amplitude resolution than those of existing systems. On a challenging public dataset with thousands of GNSS signal snapshots from real-world scenarios, the method achieves reliability and accuracy comparable to existing algorithms that require more complex and expensive hardware with higher energy consumption. The algorithms were published in the proceedings of the 2021 Conference on Embedded Networked Sensor Systems (see Table 1.1), open-sourced (see Table 1.2), and used to implement a public web service for cloud-based location estimation from low-quality GNSS signal snapshots (see Table 1.3).

1.2.2 Low-Cost Open-Source Snapshot GNSS Receiver Hardware

The aforementioned robust algorithms allowed to build a snapshot GNSS receiver as a second core component of the SnapperGPS project, which has an unmatched low cost of around $20[1], can acquire two positions per hour for more than a year on a 40 mA h battery, and is described in Chapter 4. This receiver is unique in the sense that it addresses all five points 1–5 laid out in Section 1.1. Its electronics design and firmware were open-sourced, are certified by the Open Source Hardware Association (OSHWA)[2], are described in a publication in the Journal of Open Hardware (see Table 1.1 and Table 1.2), and were independently reproduced by several third parties.

1.2.3 Evaluation of Snapshot GNSS for Various Wildlife Tracking Scenarios

The entire SnapperGPS system was deployed in several wildlife tracking field trials, where accuracy, robustness, and low energy consumption were demonstrated in practice. The results of one deployment were presented at the International Sea Turtle Symposium 2022 (see Table 1.1). As of writing, more than 200 replications of the hardware have been tested and used around the world, mostly as part of a user study. This study was the first one that assessed the potential of snapshot GNSS for wildlife tracking across a variety of different use cases and deployment scenarios and collected evidence for the five points 1–5 listed in Section 1.1.

1.2.4 A Snapshot GNSS System with Cloud-Offloading via a Narrow-Band Cellular Connection

A further contribution is a snapshot GNSS receiver with cloud-offloading via a narrow-band cellular connection for near-real-time localisation. This addresses the limitation of the basic SnapperGPS system that a receiver needs to be recovered before its data can be processed. Designing a wireless low-power snapshot GNSS receiver is challenging, though, since low-power communication networks lack the

[1] At the start of the global chip shortage in 2020 and 2021.
[2] `https://certification.oshwa.org/uk000049.html`

bandwidth for off-loading GNSS snapshots while the use of broadband networks jeopardises the power advantage of snapshot GNSS. A narrow-band cellular network occupies a unique spot in between. The power consumption of the system is evaluated in different settings. The results show that the proposed approach to data offloading is feasible and can outperform existing approaches in certain scenarios.

1.2.5 Evaluation of Smoothing of Snapshot GNSS Data

This contribution addresses a shortcoming of all published snapshot GNSS methods, including the original SnapperGPS algorithms: They basically process all signal snapshots independently. Smoothing is—if at all—only applied to the calculated position fixes in a separate second stage, usually by averaging fixes during stationary phases. In contrast, this dissertation proposes an algorithm that leverages factor graph optimisation and incremental smoothing to jointly process signal snapshots that were captured close in time. This aims at improved robustness in environments with partial sky occlusion and increased accuracy in general, while also transparently propagating measurement uncertainty from the signal snapshot to the position fix. Overall, effective smoothing algorithms extend the relevance of the snapshot GNSS technique from long deployments with low sampling rates to shorter deployments with higher rates, where accuracy is often more of a concern. In these deployments, accuracy can be close to the one of commercial non-snapshot-GNSS modules with higher energy consumption and more complex hardware.

1.2.6 Factor Graph Algorithms for Fusion of Raw GNSS Data with Other Sensing Modalities

Smoothing of GNSS data is equivalent to fusion with a motion model of the receiver. The same techniques can be used for fusion of GNSS data with other sensing modalities. In this light, the final contribution also leverages factor graph optimisation for raw GNSS data processing, but for highly accurate localisation, a requirement more frequently encountered in, e.g., robotics rather than in wildlife conservation. This dissertation proposes a robust approach that tightly fuses raw

GNSS receiver data with inertial measurements and, optionally, lidar observations for robust, drift-free, and smooth mobile localisation. Unlike other approaches to accurate localisation with GNSS, this system does not require a connection to a base station, which is a key benefit. On a public dataset, the approach achieves accuracy comparable to a state-of-the-art algorithm that fuses visual inertial odometry with GNSS data—despite the proposed approach not using the camera, just inertial and GNSS data. The algorithms were partially open-sourced and published in the proceedings of the 2023 International Conference on Robotics and Automation (see Table 1.1 and Table 1.2).

Table 1.1: Scientific papers related to this D.Phil. dissertation.

Publication	**Thesis chapters**
Jonas Beuchert and Alex Rogers. "SnapperGPS: algorithms for energy-efficient low-cost location estimation using GNSS signal snapshots". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 165–177. URL: https://doi.org/10.1145/3485730.3485931	3
Amanda Matthes, Jonas Beuchert, Alasdair Davies, Juan Patino-Martinez, and Alex Rogers. "SnapperGPS: deployment of a low-cost snapshot GNSS receiver to track loggerhead sea turtles". In: *Proceedings of the 40th International Sea Turtle Symposium (ISTS40)*. International Sea Turtle Society. 2022. URL: https://ora.ox.ac.uk/objects/uuid:c9acf083-d5e5-4265-8425-67509c5e3b9b[3]	5.2
Jonas Beuchert, Amanda Matthes, and Alex Rogers. "SnapperGPS: open hardware for energy-efficient, low-cost wildlife location tracking with snapshot GNSS". in: *Journal of Open Hardware* 7.1 (2023). URL: https://doi.org/10.5334/joh.48	4, 5.2
Jonas Beuchert, Marco Camurri, and Maurice Fallon. "Factor graph fusion of raw GNSS sensing with IMU and lidar for precise robot localization without a base station". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 8415–8421. URL: https://doi.org/10.1109/ICRA48891.2023.10161522	8

[3]Joint first author.

Herval Silva, Alasdair Davies, Jonas Beuchert, Claire Painton, and Juan Patino-Martinez. *The conservation of migratory marine vertebrates: new insights into habitat use, threats, and behaviour of sea turtles.* Submitted to the International Scientific Symposium on the Canary Current Large Marine Ecosystem (under review). 2023	5.2

Table 1.2: Open-source releases related to this D.Phil. dissertation.

Repository	**Thesis chapters**
SnapperGPS: collection of GNSS signal snapshots. `https://doi.org/10.5287/bodleian:exrp1xydm`	3.3
SnapperGPS: collection of GNSS signal snapshots 2. `http://dx.doi.org/10.5287/ora-xq5b8xva7`, `https://github.com/JonasBchrt/snapshot-gnss-data-2`	7.3
snapshot-gnss-data: utilities for reading GNSS snapshots collected with SnapperGPS. `https://github.com/JonasBchrt/snapshot-gnss-data`	3.3
snapshot-gnss-algorithms: algorithms for location estimation based on short GNSS snapshots. `https://github.com/JonasBchrt/snapshot-gnss-algorithms`	3
cold-snapshot-demo: obtain a GNSS position fix from an 11-millisecond raw GNSS signal snapshot without any prior knowledge about the position of the receiver and only very coarse knowledge about the time. `https://github.com/JonasBchrt/cold-snapshot-demo`	2.5, 2.7
snappergps-pcb: printed circuit board (PCB) design for snapshot GNSS receiver. `https://github.com/SnapperGPS/snappergps-pcb`	4.2

snappergps-pcb-2: printed circuit board (PCB) design for snapshot GNSS receiver. `https://github.com/SnapperGPS/snappergps-pcb-2`	4.2
snappergps-pcb-2-1: printed circuit board (PCB) design for snapshot GNSS receiver. `https://github.com/SnapperGPS/snappergps-pcb-2-1`	4.2
snappergps-pcb-2-2: printed circuit board (PCB) design for snapshot GNSS receiver. `https://github.com/SnapperGPS/snappergps-pcb-2-2`	4.2
snappergps-firmware: firmware for snapshot GNSS receiver. `https://github.com/SnapperGPS/snappergps-firmware`	4.3
snappergps-app: web app for snapshot GNSS. `https://github.com/SnapperGPS/snappergps-app`	3.4, 4.4
snappergps-backend: web app back-end for snapshot GNSS. `https://github.com/SnapperGPS/snappergps-backend`	3.4, 4.4
snappergps-scripts: demo scripts for post-processing SnapperGPS data. `https://github.com/SnapperGPS/snappergps-scripts`	5.2, 7
snappergps-housings: housings for a SnapperGPS PCB. `https://github.com/SnapperGPS/snappergps-housings`	5.2
snappergps-accelerometer-daughterboard: accelerometer daughterboard for a basic SnapperGPS receiver. `https://github.com/SnapperGPS/snappergps-accelerometer-daughterboard`	4.2
raw-gnss-fusion: code, data, and results for fusing raw GNSS data with other sensing modalities. `https://github.com/JonasBchrt/raw-gnss-fusion`	8

Table 1.3: Miscellaneous public output related to this D.Phil. dissertation.

Item	Thesis chapters
SnapperGPS web application. 2020-2023. URL: https://snappergps.info	3.4, 4.4
Amanda Matthes and Jonas Beuchert. "Can we locate endangered sea turtles using twelve milliseconds of noisy satellite signals?" In: *Inspired Research: News from the Department of Computer Science*, University of Oxford. Oxford, UK, Issue 19 (Winter 2021), Pages 20-21. URL: https://www.cs.ox.ac.uk/innovation/inspiredresearch/InspiredResearch-winter2021.pdf.	4, 5.2
"Tracking endangered sea turtles with hardware the size of a pound coin". University of Oxford, Oxford, UK. December 2021. URL: https://www.ox.ac.uk/news/features/tracking-endangered-sea-turtles-hardware-size-pound-coin, https://eng.ox.ac.uk/news/tracking-endangered-sea-turtles-with-hardware-the-size-of-a-pound-coin/	4, 5.2
"Helping to protect endangered sea turtles". Kellogg College, Oxford, UK. April 2023. URL: https://www.kellogg.ox.ac.uk/news/turtle-tracking-tag/, https://eng.ox.ac.uk/case-studies/helping-to-protect-endangered-sea-turtles/	4, 5.2
Jonas Beuchert. "SnapperGPS: a small, low-power, low-cost location data logger". Poster and demo. Open Hardware Summit. New York, NY, USA. May 2023.	4, 5, 6
Josie A. Peters. "University of Oxford engineering & computer Scientists are helping save sea turtles". May 2023. URL: https://twitter.com/oxengsci/status/1661290512819666944?s=20, https://www.instagram.com/p/CslIVVcIYh_	4, 5.2
Jonas Beuchert. "SnapperGPS: assembly with 3D-printed box or tray". Video. November 2022. URL: https://youtu.be/pbr400zIdqU	5.2
Jonas Beuchert. "SnapperGPS: how to deploy a sea turtle tag". Video. April 2023. URL: https://youtu.be/iAc9z95s72Q	5.2

Jonas Beuchert. "SnapperGPS: a new version - changes, usage, and light-weight packaging". Video. May 2023. URL: https://youtu.be/bDywcAplRWk | 4

1.3 Thesis Overview

The following Chapter 2 summarises the background of the work, including fundamentals and reviews of relevant literature. The subsequent chapters present the different contributions of this thesis.

As part of the SnapperGPS project, Chapter 3 introduces the fast algorithms that reliably estimate positions from SnapperGPS' low-resolution GNSS signal snapshots. It includes a real-world evaluation as well as a brief description of a public web service for cloud-based location estimation using the algorithms.

The design of snapshot GNSS receiver hardware as the second core component of the SnapperGPS project is described in Chapter 4. Design decisions are justified, different hardware variants explained, and hardware properties and performance verified. In addition, firmware design and hardware calibration algorithms are described.

The SnapperGPS system was used for wildlife tracking projects, exemplary of which are outlined in Chapter 5. This chapter also presents the results of the SnapperGPS user survey and a discussion thereof.

Chapter 6 is concerned with a snapshot GNSS receiver with cloud-offloading via a narrow-band cellular connection for near-real-time localisation. The power consumption of the system is evaluated in different controlled scenarios and compared to a baseline system.

Chapter 7 proposes algorithms to jointly process signal snapshots that were captured close in time. Different algorithms are experimentally compared with baseline algorithms on a large real-world dataset.

Chapter 8 proposes a method that tightly fuses raw GNSS receiver data with inertial measurements and, optionally, lidar observations for robust, drift-free,

and smooth mobile localisation. A real-time implementation of the approach is described and evaluated on a public dataset as well as on a variety of self-collected recordings in different environments.

Finally, Chapter 9 provides conclusions and an outlook on future research directions.

2
Background

This thesis presents specific concepts for using global navigation satellite systems to localise a GNSS receiver on Earth. The background chapter shall serve as a foundation for understanding the research presented in the rest of the thesis as well as contextualising the research topic and highlighting the existing literature in this field. Therefore, the chapter starts with the basics of satellite navigation (Section 2.1). This is followed by a discussion of the GNSS signals (Section 2.2), which are broadcasted by the GNSS satellites and received by GNSS receivers. The architecture of GNSS receiver front-ends is outlined in Section 2.3, including signal processing hardware. Next, the signal processing chain is explained in greater detail, starting with the acquisition of satellites from the received raw signals (Section 2.4).

A mathematical concept to perform receiver localisation solely based on the results from the acquisition stage is coarse-time navigation (CTN, Section 2.5). It needs only a small amount of data for a fix, thus saving energy. Coarse-time navigation is usually based on least-squares optimisation. An alternative, less popular, approach to CTN is direct positioning (Section 2.6). Both, least-squares-based CTN and direct positioning, are used to build so-called snapshot GNSS receivers, which achieve a position fix with just milliseconds (a "snapshot") of GNSS signals. Section 2.7 presents existing snapshot GNSS receiver implementations, including hardware and software.

Conventional (non-snapshot) GNSS receiver proceed after the acquisition stage with further signal processing to refine a number of common GNSS observables for each acquired satellite (Section 2.8). Using a sub-set of these observables, the most standard positioning concept is code-based positioning (Section 2.9). However, basic code-based positioning has a few disadvantages, e.g., a long time to achieve a first fix. Some of these limitations can be addressed with snapshot GNSS or, alternatively, a concept called assisted GNSS (A-GNSS), which is introduced in Section 2.10.

Code-based positioning, including CTN, direct positioning, and A-GNSS, is limited to a few metres of localisation accuracy, which may not be sufficient for some applications. Therefore, three sections of this chapter cover two standard options to improve accuracy, which are both applied to specific problems in the body of this thesis: first, carrier-based positioning (Section 2.11), and, second, fusion of GNSS observations with either a motion model (smoothing) or with other sensing modalities. In particular, the thesis investigates how to use factor graph optimisation for this purpose (Section 2.12 and Section 2.13).

A core application addressed by the research presented in this thesis is wildlife tracking, which is an application that often requires low-cost and/or low-power receivers. Section 2.14 sketches how GNSS are currently used for wildlife tracking. It is the foundation for design requirements for most of the GNSS receiver designs presented in this thesis and also establishes limitations of current approaches. Finally, Section 2.15 establishes the state of the art of tracking of aquatic animals—a field where brief signal acquisition times are crucial and, hence, snapshot receivers potentially impactful.

2.1 Satellite Navigation

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 [26, 27]. Multiple satellites—about 30 per system—orbit the

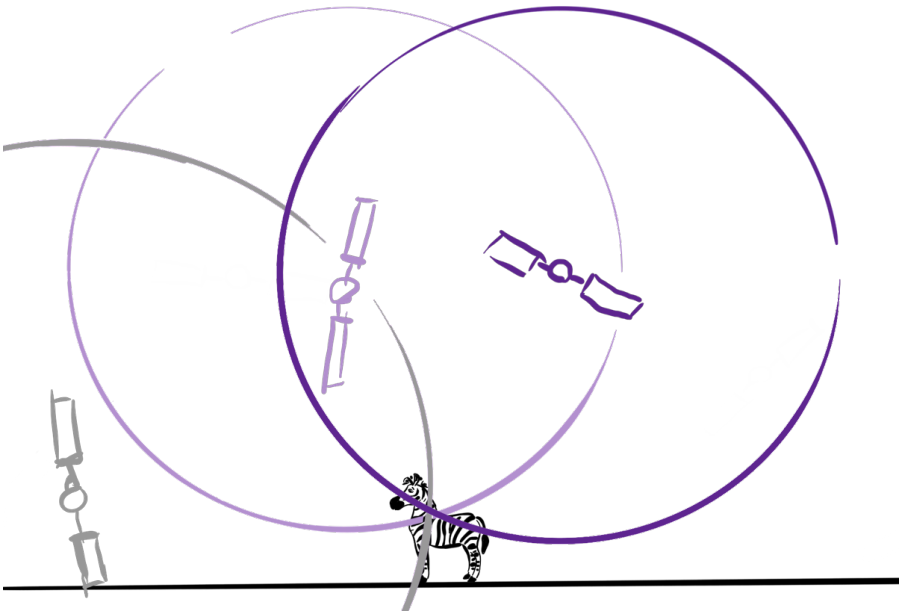


Figure 2.1: If the distances of the receiver to multiple satellites are known, then the receiver's position is found at the intersection of spheres centred at the satellites. (Any temporal dependencies are neglected for simplicity in this illustrative figure.)

Earth and transmit radio 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, cf. Figure 2.1.

2.2 GNSS Signals

Navigation satellites transmit signals in different radio frequency bands [26, 27]. Most civilian low-cost receivers operate in the GPS L1 band with a centre frequency of 1.575 42 GHz [26], which is also the centre frequency of 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 using a single-band radio front-end.

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 [26–28]:

- The sinusoidal carrier wave with a frequency of 1.575 42 GHz,

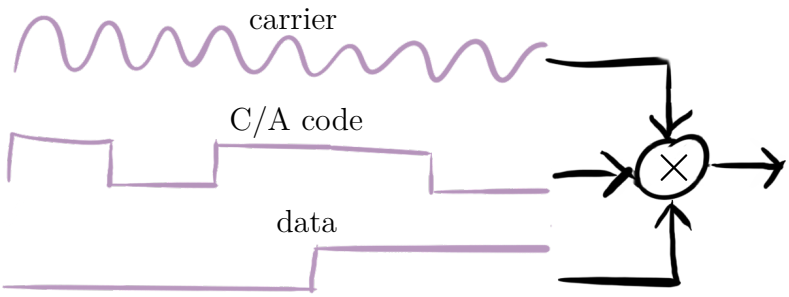


Figure 2.2: Three sub-signals form a GPS satellite's L1 signal: a sinusoidal carrier wave with 1.575 42 GHz centre frequency, the C/A code, a pseudorandom binary signal with 1.023 MHz bit rate, and the navigation data with 50 Hz bit rate. (Figure not to scale.)

- 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.

See Figure 2.2 for an illustration.

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) [27, 29–31].

2.3 GNSS Receiver Front-Ends

The signal capturing process of most types of GNSS receivers is similar. First, an antenna with an appropriate centre frequency and bandwidth captures the raw

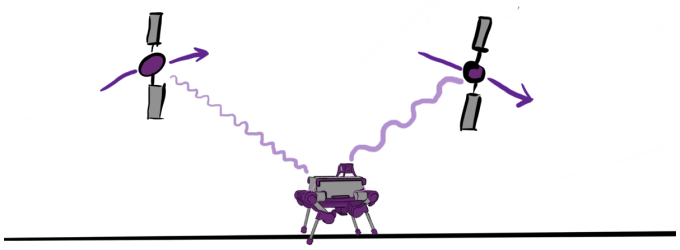


Figure 2.3: Doppler effect: the signal of an approaching satellite appears to have a higher carrier wave frequency while the receiver perceives a lower frequency from a receding satellite. This allows to estimate the relative velocities between the satellites and the receiver. If the absolute velocities of the satellites are known, then conclusions can be drawn about the absolute velocity of the receiver.

signal. For example, the carrier frequencies of the L1, E1, B1C, and L1C GNSS signals that are received by a stationary or slow-moving receiver are in the interval 1.575 42 GHz ±4.2 kHz [26, 32]. 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, cf. Figure 2.3.

Since sampling and storing data at a rate of multiple gigahertz is not possible with a low-cost receiver, an analogue circuit down-converts 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 1.575 42 GHz [26, 27], cf. Figure 2.4. 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

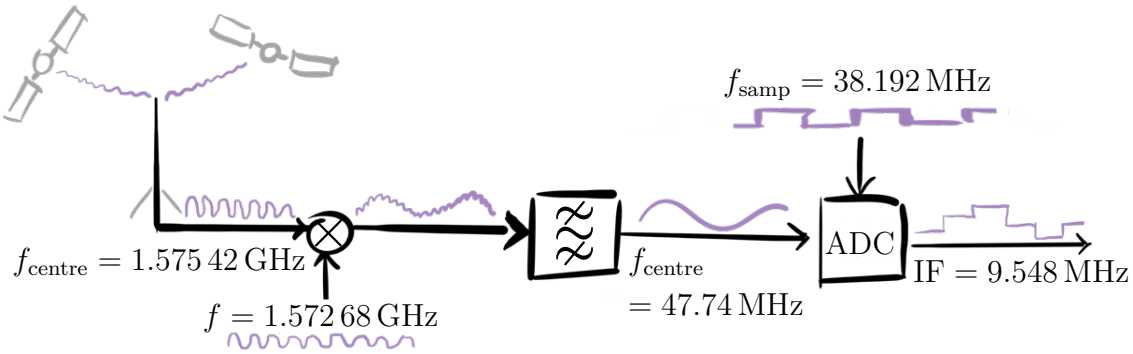


Figure 2.4: An exemplary receiver captures GPS L1 signals with a centre frequency of 1.575 42 GHz and mixes them with a locally generated sine with 1.527 68 GHz. After bandpass filtering, this results in a signal with a centre frequency of 47.74 MHz, which is subsequently digitised by an ADC with 4-bit amplitude quantisation and a sampling frequency of 38.192 MHz. Since sampling repeats the spectrum of the analogue signal periodically at the sampling frequency, this leads to a final IF of 47.74 MHz–38.192 MHz =9.548 MHz [27].

a quadrature (Q) signal, which can be interpreted as real and imaginary part of a complex IQ signal [26, 27, 33].

Finally, an analogue-to-digital converter (ADC) digitises the so-called baseband signal [26, 27]. 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 and/or process. 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.

The digital signal processing steps that follow the signal capture can either be implemented in hardware in a dedicated chip or in software on a general-purpose (micro-)processor (software-defined receiver, SDR) [26, 27, 34]. The descriptions in the following sections apply to both designs.

2.4 Satellite Acquisition

Two goals of satellite acquisition are to identify which satellite's signals are present in the captured raw signal and to determine their code phases. The code phase of a received GNSS signal is the shift between the start of a new code period and the

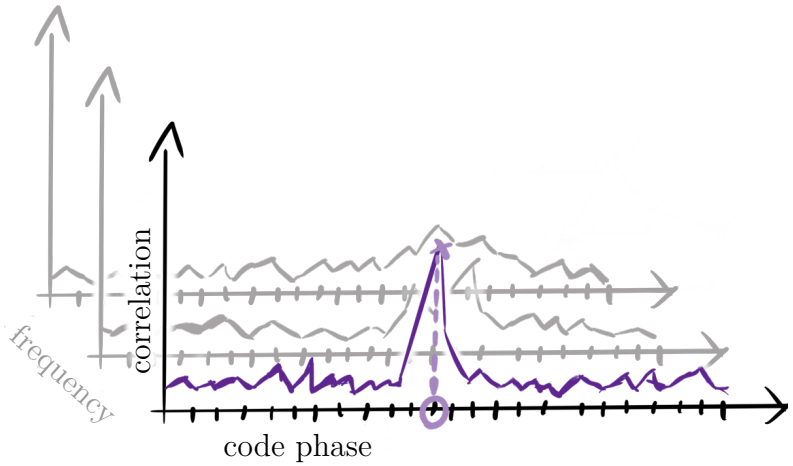


Figure 2.5: Parallel code phase search (PCPS) finds for each satellite the carrier frequency and the code phase with the highest correlation between the captured signal and the expected signal.

start of a signal recording. Code phases are the crucial ingredient of code-based positioning (Section 2.9), e.g., coarse-time navigation (Section 2.5). A popular acquisition algorithm is parallel code phase search (PCPS) [26, 27, 35]. 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 result 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 captured raw signal. Furthermore, the codes have an auto-correlation function with a single narrow peak[1] 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, cf. Figure 2.5. 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 [27], or the ratio between the highest and the second highest peak [32].

The PCPS procedure opens up a few design choices. For example, there are multiple ways to process incoming raw signals 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 incoming signal chunk 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 bit rate 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 a captured raw signal,

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

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 [32]. 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 [26]. 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 [30, 31, 36]. 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.

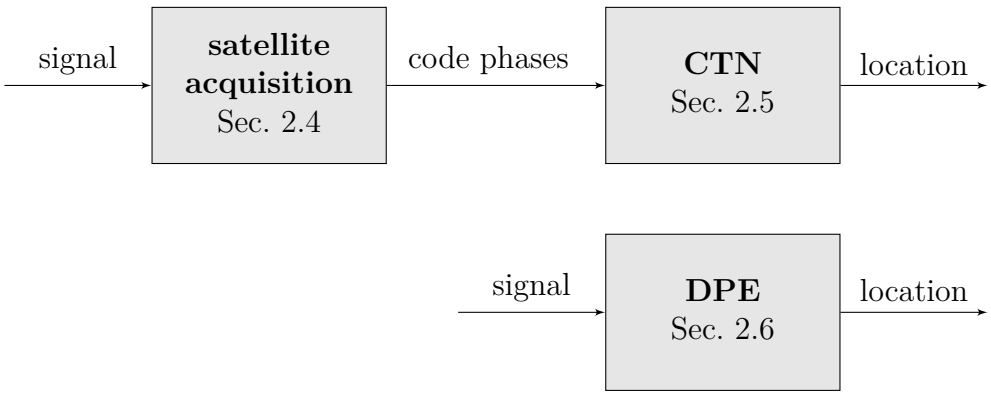


Figure 2.6: Block diagrams comparing localisation via satellite acquisition followed by coarse-time navigation (CTN, top) with direct position estimation (DPE, bottom).

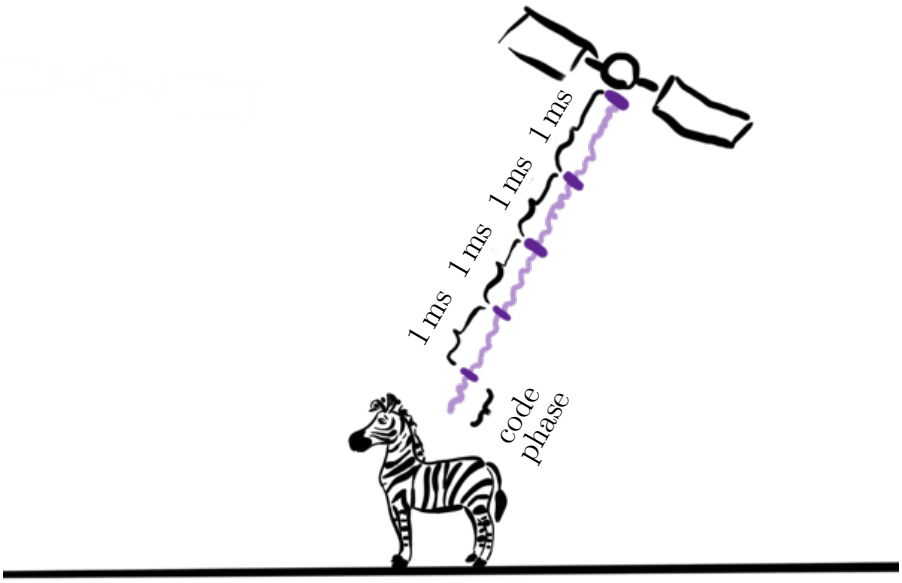


Figure 2.7: Since 1 ms signal travel time corresponds to a distance of 300 km, its sub-millisecond part—the code phase—provides enough information to reconstruct the full travel time if a coarse initial receiver position is known.

Finally, if a satellite broadcasts multiple (civil) signals in the same band, then several options exist to make use of both [31]. However, this applies to GPS only with its L1 and L1C signals. Since only six of the 31 operational GPS satellites are capable of broadcasting L1C signals in 2023, this summary skips these options.

2.5 Coarse-Time Navigation

Coarse-time navigation uses the code phases of the acquired satellites to estimate the receiver position [19], see Figure 2.6. 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, cf. Figure 2.7. 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 [19]. 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 [19].

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

$$\rho = \|\mathbf{p}_\mathrm{s}(t_\mathrm{r} + \delta t_\mathrm{r}) - \mathbf{p}_\mathrm{r}\| + b - c \cdot \delta t_\mathrm{s}(t_\mathrm{r} + \delta t_\mathrm{r}) + a(t_\mathrm{r} + \delta t_\mathrm{r}, \mathbf{p}_\mathrm{r}) . \tag{2.1}$$

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 [27, 37–39].

Usually, 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

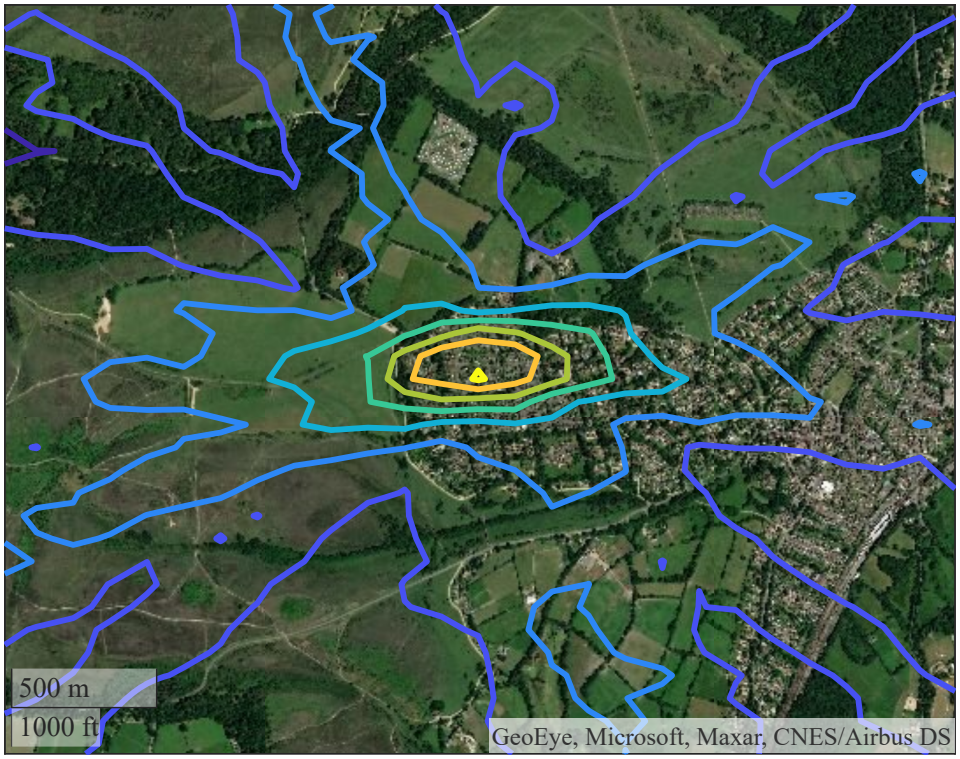


Figure 2.8: Contour plot of an exemplary likelihood of observing a single GPS snapshot given position and time.

and their corresponding predictions [19]. 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) [19, 40, 41]. The main drawback of least-squares-based 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 algorithm with reported runtimes of 0.1–0.5 s, including GPS satellite acquisition [32, 42, 43].

2.6 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, see Figure 2.6. It aims at robustifying CTN for challenging (e.g., weak) signals [44–46]. 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, cf. Figure 2.8. For the latter, Closas Gómez proposes different methods that optimise over the four-dimensional space spanned by receiver position and time [44]. 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. Section 2.5. Second, all methods are computationally expensive and optimisation over search spaces with diameters larger than a few hundred metres is infeasible.

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 [46]. 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 pre-computations 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.

2.7 Snapshot GNSS Systems

A snapshot GNSS receiver is a receiver that performs localisation purely based on brief GNSS satellite signal captures ("snapshots") [47]. The snapshots are too short to decode any payload data sent by the satellites, including timestamps and ephemerides, only satellite acquisition (Section 2.4) is possible. Therefore, CTN or DPE is required for positioning.

Snapshot GNSS is a GNSS concept, which, by design, has significantly lower energy needs, resulting in small, light-weight, energy-efficient, and potentially low-cost receivers [32, 43, 46–50]. Instead of capturing seconds or even minutes of the satellite signals for a fix, a snapshot receiver records just a few milliseconds. Since the energy consumption of a wireless receiver is dominated by the analogue blocks of the RF front-end such as amplifier, oscillator, and mixers (Section 2.3) [51], this reduces its power consumption by several orders of magnitude compared to a traditional GNSS approach. Additionally, a snapshot GNSS device does not need to calculate its position on-board, saving even more energy and lowering the requirements for its compute hardware. Instead, it can just locally store the raw signal snapshots until the deployment ends. Afterwards, the data processing can be off-loaded to the cloud [14, 32, 46, 47, 49, 50, 52].

Another advantage of snapshot GNSS receivers with cloud-offloading is that they can make use of multiple GNSS (for improved accuracy and robustness) without requiring more complex hardware or higher energy consumption [47]. This is because all GNSS-specific computations are off-loaded.

Similarly, snapshot GNSS hardware is less prone to becoming out-dated than receivers with on-board processing. If GNSS operators add new satellite signals, snapshot GNSS hardware does not need to be changed to make use of them as long as they are in the same frequency band. For example, many receivers that are in operation as of writing are limited to the traditional GPS L1 signals [2, 3, 15, 53–57] and sometimes the GLONASS G1 signals. Even more cannot handle the fairly recently (2017 [29]) introduced BeiDou B1C signal, and many are not

Table 2.1: Existing snapshot GNSS systems.

	CO-GPS	Baseband Technologies	ETH Zürich	ATS G10 Ultralite GPS	FastFix
Reference	[32]	[42]	[50]	[58]	[14, 20]
Memory	8 MBit ≤ 1,000 snapshots	4 GB ≤ 2,000,000 snapshots	2 GB 65,600 snapshots	4 GB ≤ 244,000 snapshots	2 GB[a] 244,000 snapshots
Maximum deployment duration	1.5 years (2 AA batteries, 1 fix/s)	18 days–1 year (10 mA h) weeks (coin cell) years (phone battery)	683 days (coin cell, 235 mA h, 4 fixes/h)	80 min (19 mA h, 1 fix/s) –759 days (200 mA h, 1 fix/h)	infinite (solar)
Snapshot duration	5 chunks of 2 ms	2–20 ms	1–30 ms	?	4 ms
Quantisation	2x2 bit	?	2 bit	?	1 bit @ 16.368 MHz or 2 bit @ 8.184 MHz
Sampling frequency	16.368 MHz	?	16.368 MHz	?	
Accuracy	<35 m[b]	median <9 m	<25 m[c]	mean 15.5 m[d]	median 27.7 m[e]
Weight	?	?	1.3 g	11 g	?
Size [mm]	70×52	22×27	23×14	32×23×12	?
Available 2022	no	yes[f]	no	no	no
Price	N/A	189 USD	N/A	N/A	N/A
Open source	no	no	no	no	software (partially)

[a]Flexible through use of microSD card.
[b]Data from commercial GPS front-end, not snapshot receiver.
[c]Rooftop with good sky visibility.
[d]Third-party evaluation by McMahon et al.[1].
[e]Stationary test setup.
[f]Evaluation board.

prepared for GPS L1C, which is currently being introduced, and the potential future GLONASS L1OCM signal.

Table 2.1 details pure snapshot GNSS systems that have been developed in the past. Only Baseband Technologies currently offers its solution for purchase [42]. It is available as a proprietary development board and, therefore, not readily available for deployments. Realistic real-world evaluations have been published only for the FastFix system and the ATS G10 module, but the latter was found to be unreliable due to battery and software failures [1]. Furthermore, all existing solutions also use high sample rates and/or long snapshots at multi-bit resolution. This improves the snapshot quality but also requires complex and expensive hardware. For example, the ETHZ receiver records two-bit signals sampled at 16 MHz, which limits their microcontroller choice to one with a parallel input capture interface (PARC) [50].

The CO-GPS receiver 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) [32].

Moreover, the hardware is only part of a full snapshot GNSS solution. A complete system also needs a signal processing chain to calculate positions from the raw snapshot data. However, this software is not openly available for any of these systems, which renders users dependent on the technology provider over the whole lifespan of their devices.

An exception is the system from Molteno [14, 20], which provides some of the software open-source. However, it is unclear how robust these algorithms are in practice. The author himself proposes improving robustness as potential future work. Furthermore, the difference between 60,000 collected snapshots, but only 26,000 fixes achieved, could be caused by limited robustness, although, the exact reason is unclear.

2.8 GNSS Observables

After a satellite is acquired in a certain frequency band (Section 2.4), a conventional (non-snapshot) GNSS receiver uses further base-band signal processing blocks to refine three core estimates for the satellite: pseudorange, carrier phase, and Doppler shift [26, 33, 59, 60].

The pseudorange is the observed travel time of a radio signal from the satellite to the receiver multiplied with the speed of light, cf. Figure 2.9, and is already introduced in Section 2.5. However, a GNSS module that does not employ the snapshot technique receives and decodes the broadcasted signal transmission timestamp such that the pseudorange does not need to be reconstructed from the code phase only.

A slightly different pseudorange model than Equation (2.1) that is commonly used in conventional receivers is [27]

$$\rho = \|\mathbf{p}_\mathrm{s} - \mathbf{p}_\mathrm{r}\| + c \cdot (\delta t_\mathrm{r} - \delta t_\mathrm{s} - \delta T_\mathrm{s}) + \delta\rho_\mathrm{T} + \delta\rho_\mathrm{I}. \tag{2.2}$$

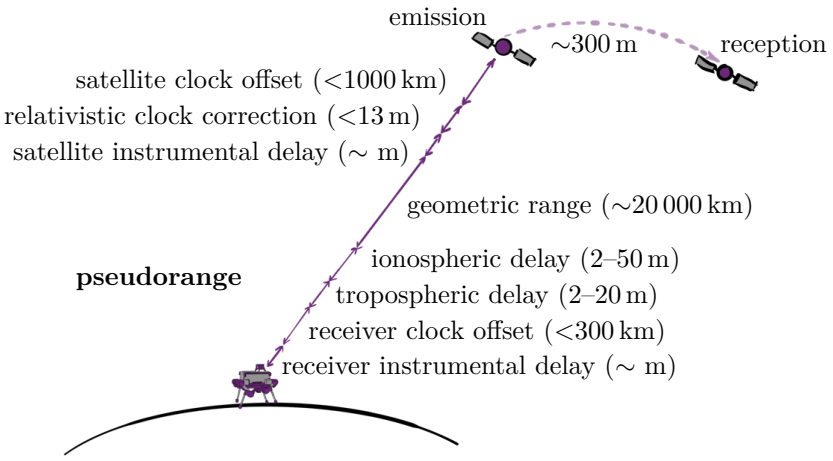


Figure 2.9: Quantities that contribute to a receiver's pseudorange measurement and their magnitudes.

In this equation, $\|\mathbf{p}_\mathrm{s} - \mathbf{p}_\mathrm{r}\|$ is the true geometric range, i.e., the spatial satellite-receiver distance. To precisely measure the travel time, transmission time and reception time must be precisely captured w.r.t. a common reference time [33]. Both, satellite and receiver, have clocks for this, but they will not be perfectly synchronised with the reference time. Therefore, Equation (2.2) includes time offsets of the receiver and the satellite clock, $\delta t_\mathrm{r} \in \mathbb{R}$ and $\delta t_\mathrm{s} \in \mathbb{R}$. They are multiplied by the speed of light c to obtain their range equivalent. Furthermore, different layers of the Earth atmosphere slow down the broadcasted signal [33]. This is modelled with a weather-dependent delay $\delta\rho_\mathrm{T} \in \mathbb{R}^+$ in the troposphere and an additional variable delay $\delta\rho_\mathrm{I} \in \mathbb{R}^+$ in the ionosphere (both summarised as a in Equation (2.1)). In addition, small instrumental delays occur on the receiver and the satellite side, respectively, when broadcasting and capturing the GNSS signal. However, the receiver instrumental delay is usually included in δt_r, such that just the satellite instrumental delay $\delta T_\mathrm{s} \in \mathbb{R}^+$ appears in Equation (2.2). Unlike the clock offsets and the tropospheric delay, the ionospheric delay and the instrumental delays depend on the signal frequency, i.e., they are different in different frequency bands. The instrumental delay also depends on the type of message (code) that the satellite transmits via the radio signal.

Even with a conventional receiver, the uncertainty of a pseudorange observation

has a standard deviation of about 1 m or more, depending on the receiver hardware [27, 33]. This is because the receiver must precisely locate the bit changes of the binary satellite codes, which occur only every 300 m, to measure the pseudoranges.

The satellites transmit their data on carrier signals in the gigahertz range, i.e., sine waves with fixed frequencies, cf. Section 2.2. If the receiver could measure the number of sine-wave periods between the satellite and itself, then this would—multiplied by the wavelength—serve as an additional observation that is proportional to the receiver-satellite distance. This measurement would be more accurate than the pseudorange because signal wavelengths are in the range 18–26 cm [33]. However, the receiver cannot count the absolute number of sine-wave periods. Instead, it can observe the phase of the carrier wave and count the change in the number of waves since the receiver first locked onto the signal. (The sum of these values is usually referred to as the *carrier phase*.) Thus, the range observation can only be inferred up to an unknown integer number $N \in \mathbb{N}$ of wavelengths $\lambda \in \mathbb{R}^+$, known as the *integer ambiguity*, which is the number of full waves when the signal was locked. Most of the terms in the carrier phase model

$$\phi = \|\mathbf{p}_\mathrm{s} - \mathbf{p}_\mathrm{r}\| + c \cdot (\delta t_\mathrm{r} - \delta t_\mathrm{s} - \delta T_\mathrm{s}) + \delta\rho_\mathrm{T} - \delta\rho_\mathrm{I} + \lambda \cdot N + \lambda \cdot \omega \qquad (2.3)$$

are the same as in the pseudorange model Equation (2.2) [33]. However, the instrumental delays of receiver and satellite (δT_s) are different for the carrier phase than for the pseudorange and also depend on the carrier frequency. Furthermore, the delay in the ionosphere $\delta\rho_\mathrm{I}$ has the opposite sign in Equation (2.3) than in Equation (2.2). There is also a small wind-up effect $\omega \in \mathbb{R}$ resulting from the interplay of the changing satellite orientation and the circularly polarised carrier wave [61]. The magnitude of $\lambda \cdot \omega$ ranges from centimetres to a few decimetres. The standard deviation of the carrier-phase uncertainty is small, only about 5 mm if all other terms are precisely modelled.

Finally, the Doppler shift is the change of the carrier frequency depending on the relative velocity of the receiver w.r.t. the satellite. Doppler shifts have already been mentioned in Section 2.3 and Section 2.4, and can also be used for satellite

navigation [43], cf. Figure 2.3. The measured Doppler shift $\delta f \in \mathrm{R}$ is related to the derivative of the pseudorange [43]

$$\dot{\rho} = -(\delta f - \delta f_0) \cdot \lambda. \tag{2.4}$$

This is a simplified model that includes a receiver frequency offset $\delta f_0 \in \mathrm{R}$. If all other terms on the right hand side of Equation 2.2 are constant, then $\dot{\rho}$ equals the derivative of the geometric range $\frac{\delta}{\delta t} \|\mathbf{p}_\mathrm{s} - \mathbf{p}_\mathrm{r}\|$, i.e., the relative receiver-satellite velocity.

2.9 Code-Based Positioning

Depending on the application scenario, different satellite navigation concepts can be used, which employ the observables from Section 2.8 to estimate the receiver position. Some have lower hardware requirements and achieve lower accuracy and others need more complicated equipment, but promise smaller positioning errors.

Table 2.2 and Figure 2.10 provide overviews of the concepts that are discussed in the following.

Code-based positioning (or standard positioning service, SPS) is a basic method and works with a stand-alone single-band receiver [26, 27, 33, 59]. It only uses the observed pseudoranges to at least four satellites from a single radio frequency band [26, 33, 59]. Employing them, the method estimates four unknown variables: the three-dimensional receiver position and the combined receiver clock error and instrumental delay [26, 27, 33]. For this, it minimises the residuals

Table 2.2: Comparison of GNSS-based positioning methods: code-based positioning (SPS, Section 2.9), differential GNSS (DGNSS, Section 2.9), real time kinematics (RTK, Section 2.11), and precise point positioning (PPP, Section 2.11).

	SPS	**DGNSS**	**RTK**	**PPP**
no. of receivers	1x rover	base + rover	base + rover	1x rover
frequency bands	single/multi	single/multi	(single/)multi	multi
precise orbits & clocks	no	no	no/yes	yes
accuracy (good cond.)	> 1 m	≈ 1 m	< 1 m	< 1 m

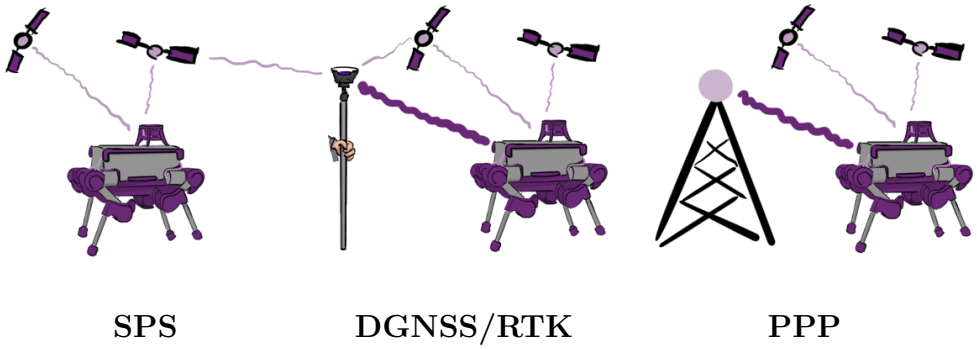


Figure 2.10: Different GNSS positioning concepts require different hardware setups. Code-based positioning (SPS, Section 2.9) works with a stand-alone receiver, differential GNSS (DGNSS, Section 2.9) and real time kinematics (RTK, Section 2.11) require a receiver and a base in spatial proximity that are connected via a network link, and for precise point positioning (PPP, Section 2.11) a single receiver is sufficient again, but it needs to be connected to the internet, e.g., via a mobile network, or, alternatively, the technique to be used in post-processing.

between the model-based pseudorange predictions (Equation (2.2)) and the observed pseudoranges [26, 27, 33]. Accurate predictions require knowledge of the satellite positions $\mathbf{p}_\mathrm{s}$ and clock errors δt_s and the atmospheric delays $\delta\rho_\mathrm{T}, \delta\rho_\mathrm{I}$ [26, 33]. The GNSS satellites itself as well as satellite-based augmentation systems broadcast parameters to approximately model $\delta\rho_\mathrm{I}$ [26]. However, they are imprecise and can cause errors up to a few metres in total. The receiver can obtain more accurate values from the internet, e.g., via an NTRIP[2] caster in real time from a network of ground stations that track the satellites or from databases for post-processing. Due to the magnitude of the measurement noise of the pseudorange observations and potentially imprecise satellite and atmospheric data, code-based positioning achieves only an accuracy of a few metres [47]. In general, more observations from more satellites increase its accuracy and reliability, but this requires careful weighting of the observations [26], e.g., based on the signal-to-noise ratio (SNR) or the satellite elevation.

The largest terms in Equation (2.2) are the geometric range $\|\mathbf{p}_\mathrm{s} - \mathbf{p}_\mathrm{r}\|$ and the receiver clock offset δt_r followed by the delay in the ionosphere $\delta\rho_\mathrm{I}$ [33]. The

[2]NTRIP is the acronym for Networked Transport of RTCM via Internet Protocol. RTCM is a common communication protocol for sending data from a secondary source to a GNSS receiver. The standard is named for the Radio Technical Commission for Maritime Services that created it.

latter ranges from a few metres up to more than 20 m [33], cf. Figure 2.9. To account for this delay, a single-band receiver must have information about the current conditions in the atmosphere. However, a multiband receiver can combine observations ρ_1, ρ_2 from bands with different centre frequencies f_1, f_2 to (almost) eliminate the ionospheric delay

$$\hat{\rho} = \frac{f_1^2 \cdot \rho_1 - f_2^2 \cdot \rho_2}{f_1^2 - f_2^2}. \tag{2.5}$$

In this way, a stand-alone receiver achieves higher accuracy without online access to precise ionosphere data [26, 33].

Equation (2.5) applies to both, pseudorange and carrier phase observations.

A common setup for satellite navigation consists of two receivers, which are configured as base and rover, respectively. The base is located at a precisely known position with good satellite visibility while the rover operates close to it, ideally in less than 100 km distance, and has a permanent network connection to the base [33]. The base transmits its GNSS observations via the network link to the rover.

In differential GNSS (DGNSS), the latter takes the difference between the base's observation and its own one for each satellite that is captured by both devices [33]. Because of the spatial and temporal proximity of the observations, the ionospheric delay δt_I and tropospheric delay δt_T, the satellite clock offset $c \cdot \delta t_\mathrm{s}$, and the satellite instrumental delay $c \cdot \delta T_\mathrm{s}$ are approximately the same and cancel out [33]

$$\Delta\rho = \mathbf{u}_\mathrm{s}^\mathrm{T} \cdot \Delta\mathbf{p}_\mathrm{r} + c \cdot \Delta\delta t_\mathrm{r} \tag{2.6}$$

Subsequently, the receiver employs a set of pseudorange differences $\Delta\rho \in \mathbb{R}$ to solve for the unknown position difference $\Delta\mathbf{p}_\mathrm{r} \in \mathbb{R}^3$ and clock difference $\Delta\delta t_\mathrm{r} \in \mathbb{R}$ between rover and base and adds the estimates to the absolute base position and base clock error, which are known. In Equation (2.6), $\mathbf{u}_\mathrm{s} \in \mathbb{R}^3$ is the unit vector from the base to the satellite.

Since most uncertain terms in Equation (2.2) cancel out in Equation (2.6), DGNSS achieves accuracies of about 1 m with just two single-band receivers.

2.10 Assisted GNSS

A traditional GNSS receiver usually requires more than one minute of data for a first fix (for which it needs to decode both, ephemerides and timestamps), and at least several seconds for subsequent fixes (if recently decoded ephemerides are available) [19]. However, powering the radio to capture these signals and then the processor to calculate the location is energy expensive, resulting in the need for large batteries with high capacity. This often makes traditional GNSS tags impractical for tracking small animals over long deployment periods, for example [1].

Assisted GNSS (A-GNSS) receivers address this issue by obtaining some of the satellite data in another way, which allows them to reduce their on-time, thereby saving energy [47]. This is done either by pre-loading ephemerides before the deployment or by regularly downloading them via another connection (e.g., cellular) [19, 47, 62]. However, pre-loading data is only possible for short deployments (roughly a month [62, 63]) and an additional download link requires more expensive hardware and is energy intensive.

Coarse-time navigation (Section 2.5) can be considered as a form of A-GNSS since the satellite navigation data is obtained by another means rather than the satellite-receiver downlink [19].

2.11 Carrier-Based Positioning

Standard code-based positioning and DGNSS use only a single type of observations—pseudoranges—, which does not allow for persistent sub-metre accuracy even if all model terms in Equation (2.2) are precisely known. The reason is the large standard deviation of the measurement noise of more than one metre. This limit can be overcome by using the low-noise carrier-phase observations for positioning based on Equation (2.3) [33, 64, 65].

In addition to the three-dimensional receiver coordinates and the receiver clock offset δt_r, this requires estimating a fifth unknown variable, the integer ambiguity N. Often, receivers fix N to a real (floating) number first and then refine it to

a true integer once more data is available, e.g., with the least-squares ambiguity decorrelation adjustment (LAMBDA) method [66].

With floating ambiguities, decimetre-level positioning accuracy is possible; with ambiguities fixed to integers, centimetres or even millimetres can be achieved [64].

Usually, the latter requires introducing a sixth unknown state variable, the deviation of the (zenith) tropospheric delay from the nominal one, which often cannot be modelled with the required accuracy [65].

To either precisely model or eliminate the other terms in Equation 2.3, two concepts are widespread, *real time kinematics* (RTK) and *precise point positioning* (PPP).

Real time kinematics requires two receivers in 10–20 km distance or less, which are configured as base and rover, like in DGNSS [67, 68]. Ideally, they are multiband receivers, but this is not strictly necessary. The rover estimates the six variables with the help of the base's observations, which need to be transmitted in real time. For this, it uses double-differencing: by combining two observations from the rover with the corresponding two observations from the base, it cancels all terms in Equation 2.2 and Equation 2.3 that are either receiver specific, but the same for two different observations of the same receiver, or satellite signal specific, but approximately the same for two receivers in spatial proximity.

After convergence (several seconds to several minutes), RTK can achieve at least centimetre-level positioning accuracy [33, 67].

In contrast to RTK, *precise point positioning* enables carrier-based positioning with a stand-alone multiband receiver without access to a dedicated base station [64, 65]. Since no base observations are available, the receiver must precisely model all terms in Equation (2.2) and Equation (2.3) that do not depend on unknown variables because they cannot be removed with double-differencing. This includes obtaining precise satellite orbits and satellite clock offsets in real time from a reliable online source, e.g., from a network of ground stations via an NTRIP link [65], instead of using the data that is broadcasted by the satellites itself, which is only accurate to about ± 1 m and a few nanoseconds.

Usually, the precise navigation data includes phase centre offsets of the satellite antennas and corrections for relativistic effects, too [65]. They can be neglected for SPS and cancel out in DGNSS and RTK, but are relevant for accurate PPP. Similarly, Earth deformation effects and an antenna phase centre offset shall be considered on the receiver side for optimal accuracy [65].

In addition, the ionospheric delay is removed by combining measurements from different frequency bands [65], cf. Section 2.9.

After convergence, which usually takes longer than for RTK, PPP achieves centimetre-level positioning accuracy in static applications and decimetre-level accuracy in kinematic ones.

2.12 Sensor Fusion and Factor Graph Optimisation

Sensor fusion is key to make localisation of mobile systems more accurate and robust in areas without or with limited sky visibility, especially, in robotics, but also elsewhere. Sensor fusion exploits the different properties of different sensing modalities. For example, inertial measurement units (IMUs) and wheel or leg odometry provide information on the relative movement of a mobile platform w.r.t. a previous state. Furthermore, cameras and lidar enable localisation in local maps using relative positioning w.r.t. features of the environment. However, none of these sensors can completely eliminate a drift of the position estimate w.r.t. a global reference frame during long-term operation in a large unknown environment. Errors accumulate and are not corrected as long as no loop-closures are detected. In consequence, the uncertainty of the position estimate usually grows with time. In contrast, positioning with GNSS provides position estimates w.r.t. a global frame, whose uncertainties are independent from the time and distance that the platform has travelled already. Therefore, a system that fuses measurements from the aforementioned local sensors can benefit from integration of GNSS for long-term autonomy in changing environments and vice-versa.

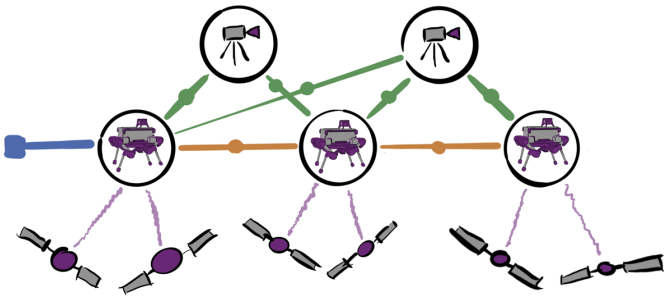


Figure 2.11: Factor graph for robot perception with variable nodes (black circles), which represent either the state of the robot at certain points in time or visual landmarks, and factor nodes, which represent either prior knowledge such as an initial robot pose (blue) or measurements such as visual features (green), inertial measurements (yellow), and GNSS observations (purple).

Sensor fusion for localisation, i.e., estimation of position states from multi-modal measurements, can be treated as probabilistic inference problem and solved with maximum a-posteriori (MAP) estimation. Factor graphs are a popular graphical model for this problem class and the de-facto standard framework for robotic navigation [69]. A factor graph represents the factorisation of the conditional probability distribution for the MAP estimation. It consists of two types of nodes: variable nodes, which represent the states to be estimated, and factor nodes, which are derived from the measurements or represent prior knowledge, cf. Figure 2.11. Efficient algorithms exist that perform incremental smoothing on factor graphs [70, 71], i.e., in every single epoch, they jointly consider past and current measurements to estimate a complete optimal trajectory of past and current states, which best explains all measurements. In contrast, traditional filtering approaches, e.g., the extended Kalman filter (EKF), marginalise past states and measurements to obtain a current state estimate only rather than a full trajectory. Factor graph optimisation (FGO) can deliver improved accuracy for highly non-linear estimation problems: it keeps re-linearising the model around all states in the factor graph, thus iteratively reducing linearisation errors for states that would have already been marginalised by an EKF [72]. The runtime of FGO can be higher than the one of an EKF for the same problem, but shorther than the runtime of a particle filter, another algorithm to solve highly non-linear estimation problems [73].

2.13 Factor Graph Optimisation for Satellite Navigation

Fusion of individual GNSS satellite observations (rather than pre-computed GNSS fixes) with proprioceptive and exteroceptive measurements in a single estimation framework has been pursued in previous work. Traditional methods for integration of GNSS and proprioception, e.g, inertial or encoder measurements, often use filter-based estimation [74, 75]. In contrast, some recent approaches from the robotics community leverage factor graph optimisation. Factor graphs are a popular estimation framework for various fusion problems in robotics and beyond [69] and it is desirable to be able to incorporate raw GNSS observations in this manner. Furthermore, the pseudorange and carrier-phase observation models Equation 2.2 and 2.3 are highly non-linear in the unknown variables, allowing to take advantage of the favourable accuracy-runtime trade-off of FGO for many problems in this class.

Most approaches use the pseudoranges provided by the GNSS receiver [72, 76–81]. A pseudorange is the observed travel time of a radio signal from a satellite to the receiver multiplied by the speed of light, cf. Section 2.8. They can be seen as range-only observations of distant landmarks; although, pseudoranges usually have much larger meter-scale uncertainties than the centimetre-scale ones of visual landmarks [81]. For example, Gong et al., Wen et al., and Cao et al. fuse pseudoranges with combinations of visual or inertial measurements to show that a tightly coupled approach can be robust in urban canyon scenarios with limited sky visibility [78, 81, 82]. They achieve mean positioning errors of a few meters. Due to their meter-scale measurement uncertainty, pseudoranges cannot be employed for centimetre-accurate localisation—only for anchoring a trajectory in a global Earth frame and eliminating odometry drift. For accurate local navigation, these approaches rely on a combination of proprioception and exteroception.

To overcome this limitation, carrier-phase observations can also be used, cf. Section 2.8. Usually, real-time kinematic positioning (RTK) is used to resolve the integer ambiguity in real time, cf. Section 2.11. However, this requires a persistent

connection between the moving GNSS receiver (the *rover*) and a nearby stationary second receiver at a known location (the *base*).

In the literature, approaches have been described that could make use of carrier phases in real time *without the need for a base station*. For example, Suzuki uses carrier phases to create factors between states at different times to obtain relative distance measurements w.r.t. past epochs [83]. The author names this method *time-relative RTK* because of its similarity to RTK—with the current observations as rover observations and a set of previous observations as observations to a virtual base station. For each continuously tracked satellite, the approach estimates the integer ambiguity using the LAMBDA method [66, 84]. If the method cannot resolve the integer ambiguity, then no factor is added to the graph. In this way, the integer ambiguity estimation is not tightly coupled with the factor-graph-based state estimation. Combining these carrier-phase factors with pseudorange and Doppler-shift factors in a single factor graph, Suzuki achieves mean positioning errors of 2–5 cm after UAV flights over 200 m or 100 s with good sky visibility and post-processing of data from a single-frequency receiver. There was no real-time evaluation of this method and it does not address fusion with non-GNSS measurements.

Lee et al. describe a second method called *sequential-differential GNSS* in which they also create differential carrier-phase factors between different states in time [85]. This approach also cancelles the integer ambiguities. It fuses the carrier phases with pseudorange, Doppler shift, visual, and inertial measurements in a multi-state constraint Kalman filter (MSCKF). However, they also use time-relative factors for the pseudoranges. They anchor the local trajectory in the global Earth frame during an initialisation phase and afterwards only use the pseudorange and carrier-phase observations for relative localisation w.r.t. previously estimated states. This limits the usefulness of pseudorange observations for reducing *long-term* drift, especially, if sky visibility is lost intermittently or very limited. They demonstrate an root-mean-square error (RMSE) of 0.32 m on a handheld dataset where sky

visibility is never interrupted and differential GNSS factors can always be created between the current and previous state.

In summary, no method has been presented yet that tightly fuses pseudoranges and carrier phases with proprioception and/or exteroception for *long-term* autonomous localisation *in real time* using a single GNSS receiver. Furthermore, to the best knowledge, tight factor graph fusion of raw GNSS data with lidar has not been addressed in the literature.

2.14 Satellite Navigation for Wildlife Tracking

A common approach to long-term animal tracking is attaching a battery-powered GNSS receiver to the animal of interest [1–13, 15]. Usually, these receivers estimate locations on-board using code-based positioning (Section 2.9) [2, 3, 5–13, 15]. These locations are either stored on-board until the end of the trial [2, 3, 5–7, 9–11, 13, 15] or they are relayed via a wireless datalink for real-time monitoring [6, 8, 12].

Since the transmission of the satellite's payload data to a receiver via the low-bandwidth satellite down-link takes a considerable amount of time, such a traditional GNSS receiver requires in practice usually more than one minute of data for a first fix, where it has to decode ephemerides and timestamps, and at least seconds for a fix for which it can reuse recently decoded ephemerides [19]. Repeatedly capturing this amount of data during a potential deployment time of several months is energy expensive and requires a bulky battery that is too obtrusive

Table 2.3: Small commercial GNSS wildlife tracking devices.

	W510 [54]	**MicroGPS+ [55]**	**TGW-4000-4 [56]**
manufacturer	Advanced Telemetry Systems	Lotek	Telonics
dimensions	50×40×10/25×25×57/28×30×55 mm	63×32×21 mm	47×24×20 mm
mass	65/115 g	25 g	47 g
price	?	?	?
fix interval	8 h	3/8 h	2/4 h
operation time	1.75–3.5 years	90 days	0.8/1.3 years

Table 2.4: Existing low-cost open-source GNSS receivers for wildlife tracking.

	Quaglietta et al. [13]	Cain et al. [2]	McGranahan et al. [3]	Foley et al. [57]	Wild et al. [15]
dimensions	65×45×28 mm	21×18 mm	?	62×38×35 mm	25×10×0.15 mm
mass	84 g	?	<300 g	240 g	1.3 g
price	790 EUR	40–65 USD	133–166 USD	126 USD	32 USD
fix interval	6 h	3 h	20 s	6 min	1–30 s
operation time	42 days[a]	16,000 fixes	109–190 h[b]	23.6 days[c]	3.6–18.2 h
deployment success	6/6	24/24	33/39	23/30[d]	6/12
fix success	68.2%	80/100/100%	"high"	99.6%	≥90%
mean error	8.9 m[d]	12.5/3.5/4.7 m[e,f]	1.8 m[e]	14.8 m	13.5 m[e]

[a]4–15 days tested.
[b]Extension to 54 days potentially possible.
[c]10 days tested.
[d]30% of the collars had intermediate failures.
[e]Stationary test setup.
[f]Different environments.

for small animals [1, 6]. Often, a maximum tag weight of only 5% of the animal's body weight is used as a rule of thumb and research suggests that for certain species this threshold should be even lower [6, 8, 15, 86]. Consensus is that tag weight should be as low as possible [6]. To summarise, traditional GNSS-based wildlife tracking devices offer unsatisfying trade-offs between the operation time on the one hand and size and weight on the other hand for certain applications. Table 2.3 lists those three parameters for popular devices. The high price is another drawback of many commercial products and puts a tight constraint on the number of devices that most researches can afford to deploy. Manufacturers usually do not advertise prices, but GNSS receivers specifically designed for wildlife tracking range from a few hundred to more than ten thousand USD [4–6]. To overcome the latter hurdle, more affordable open-source designs of GNSS-based wildlife tracking devices have been developed [2, 3, 13, 15, 16, 57], cf. Table 2.4. However, they tend to be even less energy-efficient and more bulky and also less reliable than commercial devices and might require expert electronics knowledge or tools for assembly.

Another option is to re-purpose a GNSS receiver intended for the consumer market for wildlife tracking. For example, the I-GOTU is particularly popular [5, 7, 9–11, 53]. It weights 21.5 g, is expected to operate two months providing hourly

fixes, and costs around 80 USD [53, 87]. According to Google Scholar, i-gotUs are mentioned in over 600 publications.

As an improvement of the traditional GNSS approach, A-GNSS receivers reduce their on-time and save energy by not receiving and decoding the navigation data that is part of the satellite signals, cf. Section 2.10. Instead, they obtain it by another means, e.g., it is pre-loaded before the start of the deployment or they download it from time to time via a cellular connection. However, the former limits the deployment time to a few weeks while the latter requires additional hardware and power and, therefore, increases weight and costs.

To overcome the limiting trade-off between weight and size on the one hand and operation time on the other hand, a few projects investigate an alternative GNSS concept to conventional code-based positioning for tracking of small animals: snapshot GNSS, which allows to employ receivers that are small, light-weight, energy-efficient, and potentially affordable [1, 14], cf. Section 2.7. Instead of capturing seconds or minutes of satellite signals for a fix, a snapshot receiver records just a few milliseconds, which reduces the power consumption by magnitudes. However, very few solutions are available to end-users, none of which is low-cost, see Section 2.7 and Table 2.1.

2.15 Tracking of Aquatic Animals

Oceans, lakes, and rivers cover 64% of the Earth's surface—habitats for a large variety of species. Despite the absence of permanent human population, many of these species are threatened by human activities, including climate change induced sea temperature and sea level rise, discharge of polluting liquids and particles such as plastic from shores and vessels, and fishing and mining activities [88–90]. Targeted protection of populations of aquatic species can reduce the impact of these threats and positively influence population size [91]. One such protection measure is the creation of protected areas where limits or bans are applied to human activities such as fishing, mining, shipping, and tourism [91–94]. To maximise the effectiveness of such protected areas, understanding the spatial distribution and

Figure 2.12: A surfacing sea turtle (© Matt McGee, CC BY-ND 2.0 license, `https://flic.kr/p/9H4y6h`, cropped).

movement patterns of the population under concern is crucial [93, 95]. Both can be approximated by location tracking of a representative sample of the population [90, 95]. However, tracking of aquatic animals poses unique challenges in contrast to tracking of terrestrial and avian species. For example, monitoring with cameras or through human observations is often limited by its spatial coverage due to the large sizes of habitats and long distances covered by individuals and the expenses of deploying cameras or humans (as divers or in submarines) as well as limited vision under water [93, 96]. Acoustic monitoring is a valid option and frequently used [6, 97, 98], but also limited to confined habitats and challenged by high noise levels, fouling, and too deep or too shallow water [97]. Localisation through wireless networks on Earth is usually not possible due to the absence of infrastructure on sea. Tracking with very high frequency (VHF) radios is a candidate for some species [99] but labour intense, cf. Section 2.14, and rarely used [6].

Due to the aforementioned limitations of competing technologies, satellite tracking is of interest for tracking aquatic—especially marine—species, including (a) Doppler-based systems such as Argos and (b) GNSS such as the GPS [90, 95, 98, 100]. Due to their global coverage, they do not require potentially costly deployment of infrastructure locally. Argos typically has only an accuracy of several hundred metres to several kilometres [90] and is a paid service. In contrast, even

low-cost GNSS receivers achieve accuracies of a few metres (cf. Chapter 2) and access to all civilian GNSS signals is free. However, both types of systems have the limitation that the weak satellite signals travelling ten thousands of kilometres do not penetrate water. The penetration depth (the depth at which the amplitude of the field strength of electromagnetic radiation falls to $\frac{1}{e} \approx 0.37$ of its original value) of signals in the GPS L1 band in sea water is less than 1 cm [101]. In consequence, these systems can only be used to monitor animals that surface from time to time, such as reptiles (Figure 2.12) [90, 93, 98, 100], mammals [99], and some fish that lives close to the surface like some sharks [94]. A challenge for conventional GNSS receivers is that these surfacing times can be very short, for example, regularly only 1–2 s for sea turtles [98]. In contrast, conventional GNSS receivers always need at least a few seconds for a fix and even about 30 s at least every couple of hours to download navigation data from satellites, cf. Chapter 2. The latter can be circumvented by pre-loading satellite navigation data before a tag deployment (A-GNSS, Section 2.10), but this is limited to deployments lasting a month or less [62, 63] and does not address the first problem. Snapshot GNSS is a solution approach to both problems because it always only needs a few milliseconds of sky view to collect sufficient data for a fix [90, 98].

At least one system that successfully addresses the challenge of tracking aquatic animals with brief surfacing times exists already: Fastloc-GPS is a commercial marine wildlife tracking system that may not be a pure snapshot GNSS logger, but appears to employ a version of snapshot GNSS for the aforementioned reasons [90, 98]. It appears to be a fairly popular solution with several thousand tags successfully deployed in the past [90]. However, the system is expensive with—to the best knowledge—several thousand USD per tag [90]. This is a hurdle especially for conservation work, where budgets are often tighter, the pressure to work economically higher, and required sample sizes larger than in pure research projects. According to a review of projects up until 2018, only 19% of marine turtle tracking studies cited “conservation issues” as a main reason for tracking [90].

These FASTLOC-GPS tags use so-called saltwater switches for surfacing detection, based on measurements from electrodes on the oustside of the tag [90]. This technology requires careful arrangement of the electrodes and painting of the tag with anti-biofouling compounds prior to deployment [90].

3

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

3.1 Problem Statement

A few snapshot GPS implementations have been presented in the literature so far [32, 43, 45, 46, 49, 102]. All rely either on satellite acquisition and CTN (see Section 2.4 and Section 2.5) or on DPE/CD (see Section 2.6). 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 usually 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 [32, 42, 50, 58]. However, CO-GPS and the ETHZ evaluate their positioning algorithms on data from commercial GPS front-ends rather than from their developed snapshot receivers [32, 46], 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 [32, 43, 50], 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 [32, 43, 49, 102], 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 [44–46].

- Only the ATS G10 device was evaluated for wildlife tracking, but found to be unreliable due to battery and software failures [1]. 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. It comprises open-source receiver hardware and cloud-based software for energy-efficient low-cost snapshot GNSS. This chapter addresses the core challenge 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,
- are noisy due to low-cost hardware components, i.e., have a low SNR,
- are captured with a front-end with a potentially large frequency offset, and
- are sampled with a low-cost ADC introducing quantisation errors.

The availability of such algorithms lowers the requirements for receiver hardware and reduces the amount of data that it must capture and store.

The rest of the chapter is structured as follows: Section 3.2 describes the algorithms and Section 3.3 their evaluation. Finally, Section 3.4 summarises the implementation of the algorithms in an online service before Section 3.5 provides conclusions.

3.2 Robust Algorithms

In the light of the challenge outlined at the end of Section 3.1, this section presents a novel snapshot positioning algorithm as well as improvements of existing ones that aim to significantly increase accuracy and reliability for low-cost receivers with coarse signal resolution using probabilistic signal processing. Specifically, it describes 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. The 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 the goal is to avoid a conservative selection strategy that erroneously discards inliers. This work also presents an approach to handle the considerable frequency offsets of low-cost RF front-ends that minimises computation time.

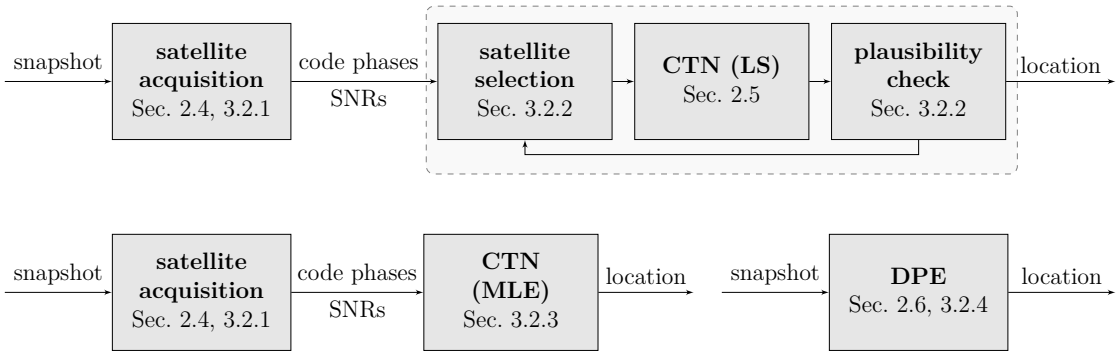


Figure 3.1: Block diagrams of SnapperGPS' alternative positioning algorithms: *LS-linear*, *LS-combo*, and *LS-SAC* (top), *MLE* (bottom left), and *DPE-[46]* and *DPE+* (bottom right).

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 [46], 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.

In the following Section 3.3, these three concepts are investigated to determine which one is best suited for ultra-low resolution signals (4 MHz sampling rate and one-bit amplitude quantisation). Implementations of all of them are provided as the first open-source snapshot GNSS algorithm compilation[1] and are accessible via a public snapshot GNSS online service[2].

In total, SnapperGPS implements eight alternative algorithms for positioning based on GNSS snapshots, cf. Figure 3.1. ***LS-single*** is a traditional approach with satellite acquisition and non-linear least-squares optimisation and ***DPE-[46]*** 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***

[1] `https://github.com/JonasBchrt/snapshot-gnss-algorithms`
[2] `https://snappergps.info`

combine the tailored acquisition stage from Section 3.2.1 with non-linear least-squares optimisation for positioning. They employ different strategies to select the satellites with reliable code-phase observations, which Section 3.2.2 presents and which have different trade-offs between robustness and runtime. Furthermore, Section 3.2.3 introduces ***MLE***, a 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, Section 3.2.4 summarises a direct positioning algorithm ***DPE+***, which is tailored to low-resolution signals, too.

3.2.1 Snapshot-Based Satellite Acquisition

The three main requirements for the tailored acquisition stage are (I) to minimise computation time, (II) to achieve high reliability and (III) to maximise accuracy of the code-phase estimates, even when 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. Section 2.4. To reduce the number of satellites to search for, the stage first predicts 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 it only considers those whose elevation is above a threshold, i.e., those which are more likely to have a line of sight towards the receiver.

Furthermore, the acquisition stage predicts the Doppler shifts for the selected satellites and only searches the frequency dimension around them.

Third, it estimates the potentially large frequency offset of each low-cost receiver front-end at the beginning of a recording and corrects the IF accordingly. If this estimate is accurate, then the acquisition routine can skip the search along the frequency axis for subsequent snapshots from the same device. Two alternative algorithms are proposed to estimate the frequency offset of the RF front-end. Both require to carry out full acquisitions with enlarged frequency search spaces

for the first few snapshots of a data record. Therefore, they are described at the end of this Section 3.2.1.

An amplitude offset of the raw snapshot due to the low-cost ADC is also expected. To avoid a DC component in the spectrum, the acquisition stage subtracts the mean of each snapshot before executing the acquisition.

For the acquisition itself, the routine splits the twelve-millisecond snapshots into individual chunks whose lengths equal the code period of the considered GNSS signal, (1 ms for GPS' L1, 4 ms for Galileo's E1, and 10 ms for BeiDou's B1C) applies zero padding if necessary, and uses 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. Section 2.4, an important feature for signals with low resolution, which addresses requirement (II).

If a GPU is accessible, the acquisition stage employs it to perform the FFTs and inverse FFTs for the correlation with NVIDIA's cuFFT library. For this, it parallelises 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 Section 2.4 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, the acquisition stage applies 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 [26]. A quadratic function is fitted to a few points around the peak and the maximum of the quadratic function is considered 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, there is interest 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. Section 2.4. The SNR is picked for this measure and calculated 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, the following part of this section returns 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 are 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. For example, if the IF is exactly zero, then a carrier frequency that differs by $-50\,\mathrm{Hz}$ from the IF is indistinguishable from one that deviates by $+50\,\mathrm{Hz}$.

The acquisition stage solves this problem using a probabilistic model and MLE. At first, it derives a prior probability $P\left(v_i = 1|\mathrm{SNR}_i\right)$ for each satellite observation $i \in 1, \ldots N$ to be reliable, i.e., to be a so-called inlier, given the associated SNR. The distribution $p\left(\mathrm{SNR}_i\right)$ of the SNRs is modelled as a Gaussian[3] 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

[3]Technically, SNRs are strictly positive while a Gaussian distribution's support includes all non-positive numbers, too. However, a Gaussian distribution is chosen because the probability contained in the distribution's tail that extends into the negative numbers is negligibly small for the considered problem in practice. In addition, efficient algorithms for interference exist for Gaussian distributions.

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

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

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

outliers. Mean, standard deviation, and prior of each component are fitted to a training dataset. This is done 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, the priors $P\left(v_i = 1 | \mathrm{SNR}_i\right) = \frac{p(\mathrm{SNR}_i | v_i = 1) P(v_i = 1)}{p(\mathrm{SNR}_i)}$ 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 are obtained.

Next, Algorithm 1 starts and predicts 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, Algorithm 1 gets the observed carrier frequency $f_i \in [f_\mathrm{min}, f_\mathrm{max}]$ from the initial acquisition runs over the enlarged search space $[f_\mathrm{min}, f_\mathrm{max}]$. To account for the ambiguity of the sign $s_i \in \{-1, +1\}$ mentioned above, Algorithm 1 creates two instances of each observation, one with the original sign $s_i = +1$ and one with the inverted sign $s_i = -1$ and assigns half the prior probability to each value in line 4. Algorithm 1 assumes 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, Algorithm 1 marginalises the sign s_i and the validity v_i in lines 8–11 and joins the individual satellites in line 13.

An unknown common offset δf_r in all observations is assumed, cf. line 2, which is estimated by maximising the output of Algorithm 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, the corresponding negative log-likelihood is minimised with a limited-memory quasi-Newton method that approximates the Broyden–Fletcher–Goldfarb–Shanno (BFGS) optimisation algorithm. The optimisation is repeated 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.

3.2.2 Satellite Selection for Coarse-Time Navigation

The SNR-induced reliability measure that is used at the end of Section 3.2.1 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 straightforward 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 Section 2.5. The latter method is referred to as *LS-single* in the following. However, choosing an appropriate threshold or satellite count is not trivial and there is always the threat that the selection process either discards a satellite that is in fact reliable or selects 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. Section 2.5. 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. The proposals in this section 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, it checks 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 the algorithm removes the satellite that is least likely to be reliable from the set and applies the least-squares algorithm again. If the solution is still not plausible, then it removes the satellite with the second smallest prior probability, too, and repeats. In this way, the algorithm iterates until it either finds a plausible solution or too less satellites are left to solve the system of equations.

The second algorithm (*LS-combo*) works similar. However, before it removes a second satellite from the set of potentially reliable satellites, it first adds the satellite that has been removed first back to the set and removes just the second least likely reliable one. In this way, it iterates through all subsets of the initial set of reliable satellites. The algorithm starts with the full set, continues with all subsets that are by one satellite smaller, and then with all subsets that miss two satellites, and so on. It aborts this combinatorial search as soon as it finds a first plausible solution. *LS-combo* has the disadvantage of a large runtime when many outliers exist since it falls into the complexity class $\mathcal{O}(N^{\frac{N}{2}})$ 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 [103], adapted to the specific problem and its non-linear character, and comprises the steps listed in Algorithm 2. It differs from RANSAC by not randomly selecting outliers. Instead, it chooses the most likely combinations of satellites guided by the SNRs of their signals. It

Algorithm 2 *LS-SAC*

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

1: Define a small number of code phases that is at least large enough to solve the least-squares optimisation in theory.
2: Find all code-phase subsets with this number of elements.
3: **loop**
4: **for all** subsets that have not been tested **do**
5: 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. *In Detail* box.
6: **end for**
7: Select the subset with the highest probability.
8: Apply CTN via non-linear least-squares as described in Section 2.5 to calculate the solution given this subset.
9: Check if the solution is plausible as described for *LS-linear*.
10: **if** the solution is plausible **then**
11: Predict all other code phases (that are not part of the selected subset) based on the solution.
12: Determine all code phases that differ by less than a threshold from their corresponding predictions.
13: Update the solution using the original small subset of code phases and those additional ones for a further least-squares step.
14: **return** the updated solution.
15: **else**
16: **if** a termination criterion is fulfilled **then**
17: **return** that the algorithm failed.
18: **end if**
19: **end if**
20: **end loop**

also does not assign the same outlier probability to all points. Instead, it uses SNR-based prior probabilities.

Like for RANSAC, the algorithm's goal is to quickly select a large set of reliable satellites for CTN by testing only small subsets and then 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, the algorithm starts with an initial subset size of six satellites to account for a potential unfavourable geometry of the satellite constellation. If it cannot find a plausible solution this way, then it repeats the algorithm with a subset size of five.

The probabilistic nature of *LS-SAC* allows 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.

In Detail: 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

$$P(V_1 = 1 \mid \mathrm{SNR}_1, \dots \mathrm{SNR}_N) = \prod_{i \in S} P(v_i = 1 \mid \mathrm{SNR}_i), \tag{3.1}$$

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

$$S_1 = \underset{S \in \mathcal{P}(\{1,\dots N\}), |S| = M}{\operatorname{argmax}} P(V_1 = 1 \mid \mathrm{SNR}_1, \dots \mathrm{SNR}_N) \tag{3.2}$$

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., $V_1 = 0$ under the assumption 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.,

$$\begin{aligned} &P(V_2 = 1 \mid \mathrm{SNR}_1, \dots \mathrm{SNR}_N, V_1 = 0) \\ &\quad = \prod_{i \in S} P(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0). \end{aligned} \tag{3.3}$$

In a general iteration k, this becomes

$$\begin{aligned} &P\left(V_k = 1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1 = 0, \ldots V_{k-1} = 0\right) \\ &\quad = \prod_{i \in S} P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right). \end{aligned} \tag{3.4}$$

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, it is desirable 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

$$\begin{aligned} &P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \\ &\quad = P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 0\right) \\ &\qquad \cdot P\left(V_{k-1} = 0 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \\ &\qquad + P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\ &\qquad \cdot P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \\ &\quad = P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right) \\ &\qquad \cdot \left(1 - P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right) \\ &\qquad + P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\ &\qquad \cdot P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right) \end{aligned} \tag{3.5}$$

$$\begin{aligned} \Leftrightarrow\ &P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 1\right) \\ &\quad = \left(\left(P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right.\right. \\ &\qquad - P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\ &\qquad\quad \left.\left. \cdot\ P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right)\right. \\ &\qquad \left./ \left(1 - P\left(V_{k-1} = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right)\right)\right. . \end{aligned} \tag{3.6}$$

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

$$\begin{aligned} &P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1\right) \\ &= \begin{cases} 1, & \text{if } s \in S_{k-1}, \\ P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0, V_{k-1} = 1, s \notin S_{k-1}\right), \\ \qquad \text{otherwise}, \end{cases} \\ &\approx \begin{cases} 1, & \text{if } s \in S_{k-1}, \\ P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right), & \text{otherwise}, \end{cases} \end{aligned} \tag{3.7}$$

where the second case is approximated such that substitution in Equation (3.6) gives

$$\begin{aligned} &P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-1} = 0\right) \\ &\approx \begin{cases} \frac{P(v_i=1 \mid \mathrm{SNR}_i, V_1=0, \ldots V_{k-2}=0) - P(V_{k-1}=1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1=0, \ldots V_{k-2}=0)}{1 - P(V_{k-1}=1 \mid \mathrm{SNR}_1, \ldots \mathrm{SNR}_N, V_1=0, \ldots V_{k-2}=0)}, \\ \qquad \text{if } i \in S_{k-1}, \\ P\left(v_i = 1 \mid \mathrm{SNR}_i, V_1 = 0, \ldots V_{k-2} = 0\right), \quad \text{otherwise}, \end{cases} \end{aligned} \tag{3.8}$$

which is the desired update equation. With Equation (3.8) and Equation (3.4), Algorithm 2 can recursively recalculate the probability for each subset to contain only inliers given all available information in line 5.

No matter which algorithm the acquisition stage uses for the satellite selection, the least-squares optimisation always provides an uncertainty measurement for each solution. Specifically, the horizontal dilution of precision (HDOP), a measure of the influence of the satellite constellation geometry on the horizontal uncertainty, is of interest. 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 [19]

$$\begin{aligned} \mathbf{Q} &= \left(\mathbf{H}^{\mathrm{T}}\mathbf{H}\right)^{-1} \\ \mathrm{HDOP} &= \sqrt{\mathbf{Q}_{11} + \mathbf{Q}_{22}}. \end{aligned} \tag{3.9}$$

The HDOP is multiplied 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.

3.2.3 Code-Based Positioning via Maximum-Likelihood Estimation

As a robust alternative to CTN via least-squares, this section introduces 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), Algorithm 3 begins 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. Section 3.2.1 and 3.2.2. Next, it starts 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 predicts the corresponding pseudoranges $\hat{\rho}_1, \dots \hat{\rho}_N$ using Equation (2.1) in line 2. However, depending on the desired accuracy, it is possible to neglect some of the terms, e.g.,

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 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, the algorithm converts 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, it is assumes 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. The algorithm also maps the difference between code-phase observation and prediction to the interval $\left[-\frac{T_i}{2}, +\frac{T_i}{2}\right]$. The final likelihood $p\left(\phi_i | \hat{\rho}_i, \mathrm{SNR}_i\right)$ for observing a certain code phase is obtained 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, the logarithms of the likelihoods are summed instead for numerical reasons and provide the log-likelihood, which has its maximum at the same location. This function is maximised 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 Section 3.2.1, multiple iterations are performed in each of which the assumed standard deviation

$\sigma_\rho \in \mathbb{R}^+$ of the valid observations is reduced to produce a solution with greater confidence, which then initialises the next iteration.

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 it can easily obtain an initial position and an initial coarse time error from the previous fix of the track, there is no straight-forward way to obtain a good initialisation for the common bias. Therefore, the following paragraph introduces a modified initial MLE run over just the common bias. For the search space of the bias, it must take into account that the common bias has a different range for different GNSS signals depending on their code period, cf. Section 2.2. 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, the search space is constrained to the interval between zero and the distance corresponding to the least common multiple of the present code periods, e.g., 20 ms if E1 and B1C signals are considered. It does not matter that the resulting common bias can overshoot the individual code periods because the modulo operator is applied when turning pseudorange predictions into code-phase predictions. The algorithm then coarsely discretises the common bias search space and calculates the log-likelihood that was introduced above for an initial position and an initial coarse time error and all elements of the bias search space. Finally, the algorithm picks the common bias that maximises the (log-)likelihood and starts the actual five-dimensional MLE.

The whole algorithm is denoted as *MLE*. This section also proposes *LS-SAC/MLE*, a combination of *LS-SAC* and *MLE* that applies *LS-SAC* first, then predicts the uncertainty of the solution as in Section 3.2.2, and proceeds with *MLE* to compute an alternative solution if and only if the predicted uncertainty exceeds a predefined threshold.

3.2.4 Direct Positioning

As a baseline for a one-step alternative to acquisition followed by positioning, SnapperGPS implements Bissig et al.'s branch-and-bound optimisation algorithm for DPE/CD (named *DPE-[46]*), which is the only existing algorithm of this class that promises robust positioning at acceptable runtimes, cf. Section 2.6. SnapperGPS also ships a novel version *DPE+* that introduces a few changes and new features for enhanced performance with low-quality signals. Unlike Bissig et al. [46] and more like Closas Gómez [44], *DPE+* derives its 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

$$\log\left(p\left(\mathbf{y}|\mathbf{p}_\mathrm{r}, b, \delta t_\mathrm{r}\right)\right) \approx \sum_{i=1}^{N} \left(\mathrm{corr}\left(\mathbf{y}, \mathbf{r}_i\left(\mathbf{p}_\mathrm{r}, b, \delta t_\mathrm{r}\right)\right)\right)^2, \tag{3.10}$$

where $\mathrm{y} \in \{-1, +1\}^K$ is the captured signal and $\mathbf{r}_i \colon \mathbb{R}^5 \to \{-1, +1\}^K$ 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 *DPE+* predicts based on the receiver position, the common bias, and the coarse time error as in line 2 of Algorithm 1 and line 2 of Algorithm 3. For the prediction of the code phase, *DPE+* allows for either pseudorange approximation or linearisation like it is the case for the two-step MLE in Section 3.2.3 to enable different accuracy vs. runtime trade-offs. Furthermore, *DPE+* processes 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. (E.g., for a 12 ms snapshot, their algorithm runs at least twelve times as long as for a 1 ms snapshot, while *DPE+*'s runtime increases only slightly.) Finally, *DPE+* 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 Section 3.2.1, *DPE+* non-coherently integrates over both E1 channels.

3.3 Evaluation

To assess the algorithm performances under realistic conditions, large GNSS data collections with thousands of snapshots from various scenarios are used. Section 3.3.1 introduces the datasets, Section 3.3.2 describes the evaluated algorithms and the conducted experiments, and Section 3.3.3 presents the results, which Section 3.3.4 discusses.

3.3.1 Data

The first data collection that is used in the experiments was recorded in 2018 and made available to the public by Watson et al. [104][4]. 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. A snapshot is extracted every 10 s such that 366 are obtained in total. To render the experiments in this chapter more challenging, they are low-pass filtered and down-sampled to 4.092 MHz, a quarter of the original rate, and only the in-phase (I) component is used. With the latter alone, the SNR is expected to degrade by 1.4 dB [26, 105]. Finally, the robust frequency offset estimation algorithm from Section 3.2.1 is used for a rough estimate (± 25 Hz) of the receiver front-end's frequency error and the IF is corrected accordingly. Coarse estimates of the receiver clock errors are obtained and the timestamps are adjusted, too.

The second data collection[5] was established in 2020 and 2021 using three SnapperGPS low-cost receivers with small Siretta Echo 27 antennas, cf. Chapter 4. It consists of four static and seven dynamic tests under various dynamically changing conditions with 3700 snapshots in total. The 225 static

[4] https://bit.ly/2vybpgA
[5] https://doi.org/10.5287/bodleian:exrp1xydm

Table 3.1: Algorithm parameters, the algorithms for which they are valid, and their values. The latter are tuned 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 Section 3.3 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-[46]*	**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

snapshots were captured 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, the frequency offsets of every test receiver are estimated based on five to ten snapshots from the start of a recording (until there is confidence that the algorithm has estimated the offsets down to ±25 Hz). They are subtract from the nominal IF.

3.3.2 Experiments

This chapter's main Experiment 1 is to apply the eight algorithms from Section 3.2 with the parametrisations in Table 3.1 to the SnapperGPS data collection using all GNSS that broadcast signals with 1.575 42 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, this particular algorithm is limited either to GPS and Galileo (Experiment 2) or to GPS only (Experiment 3). Results with the other four algorithms are obtained for those scenarios, too, for comparison. Finally, the algorithms are also applied to Watson et al.'s data in Experiment 4 to get insights on the impact of using low-cost hardware in comparison to standard equipment. Only GPS is considered for Experiment 4 because the numbers of Galileo and BeiDou satellites were comparatively small in 2018.

For all experiments, each snapshot is processed independently. Improved positioning accuracy is expected with smoothing [49], but the results of this chapter shall be transferable to scenarios where snapshots are captured at much longer time intervals, e.g., one hour, and the receiver moves up to several kilometres per hour[6]. Each algorithm run is initialised with ground truth position and time perturbed by random uniformly distributed errors and a common bias of zero. The random initial error is limited to 10 km in latitude and longitude, respectively, vertically to 100 m, and to 1 s in time.

Experiment 1, 2, and 3 are run 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. No GPU is used, which would reduce the runtime for any satellite-acquisition-based algorithm.

3.3.3 Results

This section's main measure to assess algorithm performance is the horizontal positioning error w.r.t. the ground truth tracks[7]. Figure 3.2 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 every snapshot is processed independently, there are no qualitative differences between the results of the static

[6] A separate evaluation of smoothing for snapshot GNSS follows in Chapter 7.

[7] 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. Section 3.4.

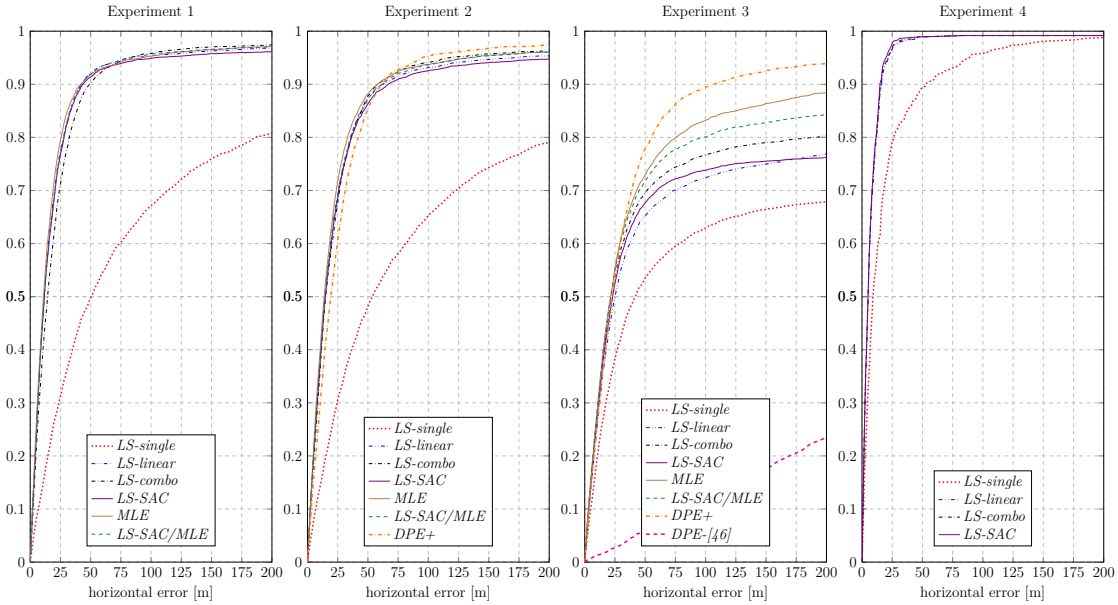


Figure 3.2: Empirical cumulative distribution functions of the horizontal localisation errors.

Table 3.2: Selected performance measures.

median horizontal localisation error [m]

experiment	data	signals	*LS-single*	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+*	*DPE-[46]*
1	[14]	L1, E1, B1C	50.2	11.6	14.4	11.7	11.0	11.7	-	-
2	[14]	L1, E1	53.7	14.4	15.1	14.8	13.9	14.6	18.7	-
3	[14]	L1	42.0	25.0	22.3	23.1	21.4	21.6	22.5	468.0
4	[4]	L1	9.0	5.1	5.1	4.9	-	-	-	-

portion of horizontal errors < 200 m

experiment	data	signals	*LS-single*	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+*	*DPE-[46]*
1	[14]	L1, E1, B1C	81%	97%	97%	96%	97%	97%	-	-
2	[14]	L1, E1	79%	95%	96%	95%	96%	96%	97%	-
3	[14]	L1	68%	77%	80%	76%	88%	84%	94%	24%
4	[4]	L1	99%	100%	100%	99%	-	-	-	-

mean algorithm runtime [s] per snapshot

experiment	data	signals	*LS-single*	*LS-linear*	*LS-combo*	*LS-SAC*	*MLE*	*LS-SAC/MLE*	*DPE+*	*DPE-[46]*
1	[14]	L1, E1, B1C	0.51	0.53	0.66	0.52	1.41	0.54	-	-
2	[14]	L1, E1	0.11	0.13	0.24	0.11	0.59	0.13	131.18	-
3	[14]	L1	0.04	0.05	0.21	0.05	0.34	0.09	35.60	842.77
4	[4]	L1	0.02	0.02	0.05	0.02	-	-	-	-

and dynamic tests, and a single diagram combines them 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 Table 3.2. Furthermore, the portion of errors smaller than 200 m is chosen 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 Table 3.2.

As third performance measure, Table 3.2 considers the mean algorithm runtime. 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.

3.3.4 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, the 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 this chapter's algorithms being a magnitude faster for GPS than methods from the literature is that they 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 Section 3.2.4 to Bissig et al.'s algorithm *DPE-[46]* 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.

Digression: DPE/CD on a GPU
Bissig et al. postulate that their *DPE-[46]* algorithm can be significantly accelerated by using a GPU as compute unit [46]. To leverage the benefits of a GPU, parallelism is crucial. However, it is unclear how this could be achieved efficiently for *DPE-[46]*. In theory, calculating the likelihood for a batch of points rather than for one after the other is an option. However, the core

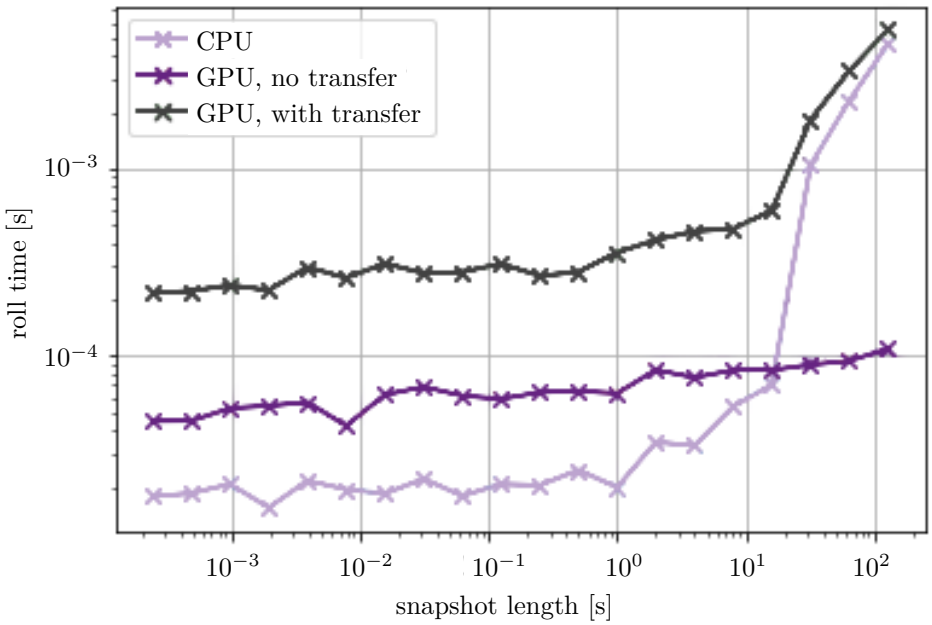


Figure 3.3: Execution time for the core array-shifting operation of DPE depending on the snapshot length. Computations either on a CPU or on a GPU (including/excluding time needed for data transfer to the GPU).

idea of *DPE-[46]* is its branch-and-bound approach. So, it is unclear which point to process next before the likelihood of the previous point has been evaluated. This can be addressed by attempting to predict the next points and process them in parallel, but this is likely to jeopardise the efficiency of the algorithm.

The main time bottleneck of *DPE-[46]* is circularly shifting large arrays and then summing them in Equation (3.10). Figure 3.3 shows measured execution speeds of the shifting operation using `numpy.roll()` on a CPU and `cupy.roll()` on a GPU (Intel® Core™ i7-9750H with 2.6 GHz and a NVIDIA GeForce GTX 1650 with 1.4–1.7 GHz, respectively.) For an integration length of 1 ms, the CPU clearly outperforms the GPU. For snapshot lengths of more than about 10 s (equivalent to 10,000 snapshots with 1 ms integration length), the CPU starts becoming slower than the GPU. However, if the data transfer times to and from the GPU are factored in, too, a speed-up is still not achieved with the examined setup.

To conclude, DPE based on branch-and-bound optimisation cannot be accelerated significantly by running it on a GPU without fundamental changes to the algorithm. The amount of computations per data point is too small to outweigh the data transfer overhead. Even with setups with faster transfer speeds, the GPU approach is unlikely to provide significant performance gains while increasing costs.

Digression: Snapshots Shorter than 12 ms
The evaluation in Section 3.3 considers only 12 ms GNSS signal snapshots. This is motivated by the choice of the MCU of the receiver, which has 8 KB of RAM. A 12 ms recording with one-bit quantisation takes up about 6 KB of this, leaving 2 KB for variables necessary for the firmware logic. But how would the SnapperGPS algorithms from Section 3.2 perform if only shorter snapshots would be available?
Executing Experiment 1 with only the first 8 ms of the same snapshots increases the median horizontal localisation errors by 1–2 m and reduces the portions of horizontal errors below 200 m by about 1%, depending on the algorithm.
With 4 ms snapshots, the performance drop is larger and varies more between the algorithms. The median horizontal localisation errors increase to at least 20 m (*MLE*) and the portions of horizontal errors below 200 m drop to 91% in the best case (*LS-combo*).
To summarise, it is possible to use shorter snapshots with SnapperGPS. The performance drops when using 33% less data are small. They are more significant when using 67% less data, but may still be acceptable for scenarios with regularly good view of the sky and where median accuracies of 20 m are tolerable. However, there is an incentive to use snapshots as long as practical to maximise performance.

3.4 Web Application

The previously introduced and evaluated algorithms are the core of the back-end of the SnapperGPS web application[8]. A corresponding open-source front-end[9] is written in JavaScript and functions as an interface between SnapperGPS receiver, user, and back-end, cf. Figure 3.4 and Figure 3.5.

The web app exposes an *upload view* to transfer recorded snapshots and timestamps from a connected device to a server after a deployment.

Following an upload, the core tasks of the back-end are:

1. To pull satellite navigation data, i.e., ephemerides and almanacs, from a public NASA[10] or BKG[11] database and

2. To calculate locations for uploaded snapshot sets.

[8] `https://snappergps.info`
[9] `https://github.com/SnapperGPS/snappergps-app`
[10] `https://cddis.nasa.gov/`
[11] `https://igs.bkg.bund.de/`

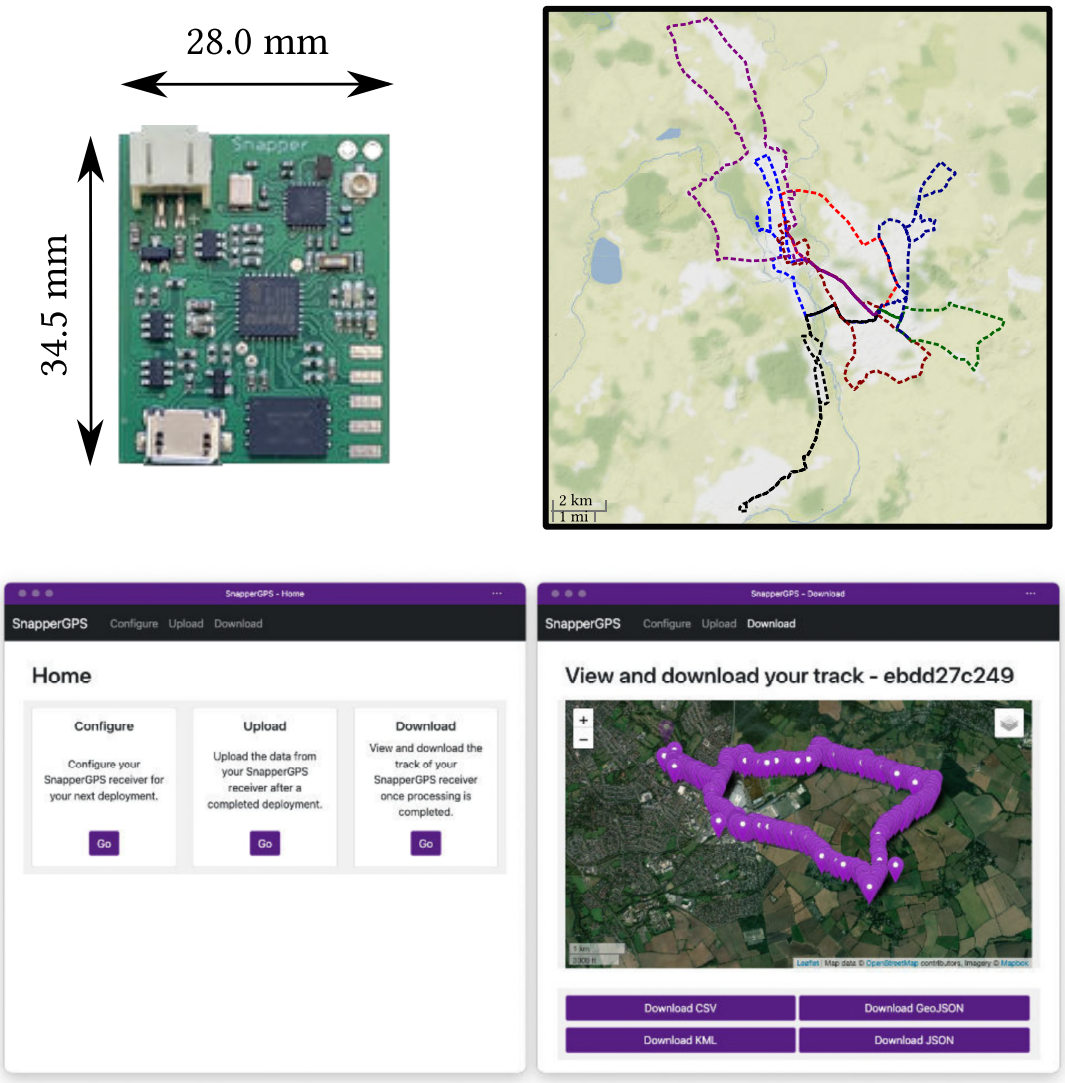


Figure 3.4: A SnapperGPS board V0.2.2, ground truth tracks of dynamic SnapperGPS tests, and exemplary views of the web application.

Task (2) is done by a custom Python back-end[12] that does not use any external GNSS libraries and is fully MIT-licensed. 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 Section 3.2.

Finally, a *download view*[13] shows the track on an interactive map and offers

[12] `https://github.com/SnapperGPS/snappergps-backend`
[13] See `https://snappergps.info/view` for exemplary tracks calculated with SnapperGPS.

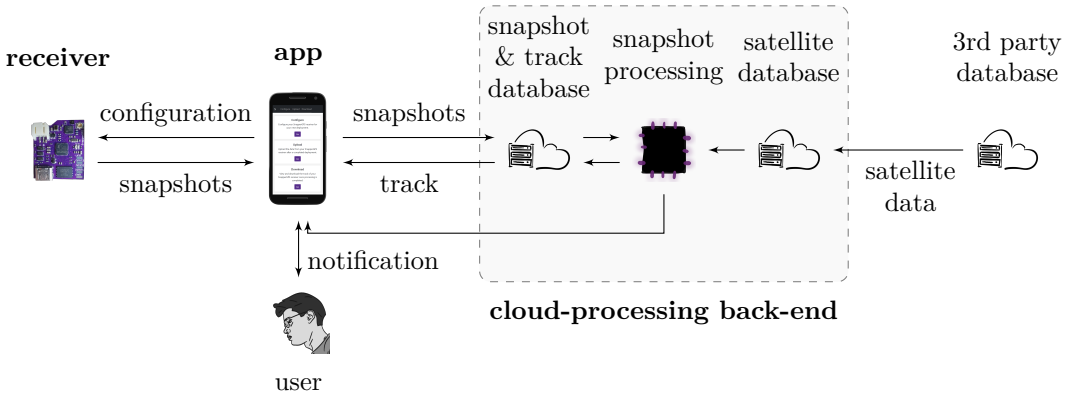


Figure 3.5: The SnapperGPS system comprising receiver, web application, and cloud-processing back-end.

downloads in various file formats after the server calculated the positions for a set of snapshots.

Digression: Use of Non-GNSS Observations for Localisation
In addition to GNSS snapshots, SnapperGPS can optionally make use of other types of observations, which are briefly presented in the following. Specifically, it can incorporate range-like observations other than satellite-receiver distances. This can be an observation of the altitude of the receiver, which is in the GPS context usually represented as the height w.r.t. to a rough ellipsoidal approximation of the Earth, the WGS84 ellipsoid [28, 106], and not w.r.t. a more accurate geoid, which would be required for height indications w.r.t. mean sea level. In consequence, obtaining a height observation w.r.t. the WGS84 ellipsoid from a digital elevation model comprises two steps and requires two models: At first, a geoid model is needed, for example, either the EGM96 [107] or the EGM2008 [108]. The model is queried at a certain latitude and longitude to obtain the geoid height w.r.t. the WGS84 ellipsoid. Second, a digital elevation model is loaded, either the ETOPO1 [109] or the SRTM1 [110], and the elevation w.r.t. mean sea level at the same coordiantes is extracted. The sum of both values is the height w.r.t. WGS84 and can be used like an additional pseudorange observation by all algorithms that are presented in Section 3.2, potentially, with a different weight/variance than the actual pseudorange observations.
Another option to obtain an additional range-like observation is to measure and record the ambient pressure and temperature with a barometer and a thermometer when capturing a GNSS signal snapshot. A relative height estimate w.r.t. a reference position can be derived from the measured pressure, the measured temperature, and a known pressure at the reference position with the hypsometric formula. The reference position can be a weather station that publishes its observations or a previous position of the same device.

SnapperGPS can also use temperature and pressure measurements to more accurately account for the GNSS signal propagation delay in the troposphere when predicting pseudoranges, cf. Equation (2.1). In general, the SnapperGPS back-end implements three different correction models for the propagation delay in the troposphere according to Goad and Goodman [38], Hopfield [60], and Tsui [26], only the first two of which account for temperature and pressure. In addition, the back-end offers two models for the ionospheric delay according to Klobuchar [39] and Tsui [26], which both use the ionosphere parameters from the GPS almanac.

3.5 Conclusions

This chapter presented tailored algorithms that enable reliable location estimation from short GNSS signal snapshots with ultra-low resolution. They allow everyone to build energy-efficient GNSS receivers at unmatched low costs, which is ideal for applications like wildlife, fitness, or certain logistics tracking. A real-world evaluation of the algorithms with moving low-cost receivers in diverse environments showed that the outlier detection methods presented in this chapter 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 is employed. This accuracy is sufficient for many applications, e.g., wildlife tracking, and the novel MLE has an even better one of 11 m. Its runtime is three-times higher, but still low. Finally, this chapter's 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 [46], which can only calculate fixes for 24% of the SnapperGPS data. In general, this evaluation is the first performance comparison of different snapshot GNSS algorithms on the same data, which is available to the public as the first open collection of thousands of GNSS signal snapshots[14].

The developed algorithms are open-source[15] and accessible via a public snapshot GNSS online service[16].

[14] https://doi.org/10.5287/bodleian:eXrp1xydM
[15] https://github.com/JonasBchrt/snapshot-gnss-algorithms
[16] https://snappergps.info

4

Open Hardware for Energy-Efficient Low-Cost Wildlife Location Tracking with Snapshot GNSS

4.1 Problem Statement

Chapter 3 presents snapshot GNSS positioning algorithms that operate on short low-resolution satellite signal snapshots. The algorithms are not designed for data from a specific hardware platform. The snapshots can be captured with any RF front-end that has an appropriate centre frequency, bandwidth, and gain, for example, a GNSS antenna connected to an off-the-shelf software-defined radio (SDR). However, a hypothetical electronic device solely designed for capturing and storing low-resolution GNSS signal snapshots could offer cost, weight, size, and energy savings in contrast to an SDR designed for general purpose use. This has been addressed as part of the CO-GPS project by Liu et al. [32] as well as by Eichelberger et al. [50], see Section 2.7. However, their hardware solutions are not particularly optimised for low costs. For example, they require relatively high sampling frequencies of at least 8 or 16 MHz and several bits for the amplitude quantisation when digitising the captured snapshots, both of which increase the requirements for the on-board hardware. These are the main hurdles to overcome on

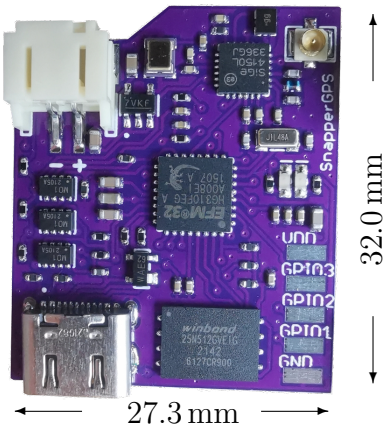


Figure 4.1: Top view of an assembled SnapperGPS receiver V1.0.0 for use with a LiPo battery and an external active antenna. No components are placed on the back of the PCB to simplify manufacturing.

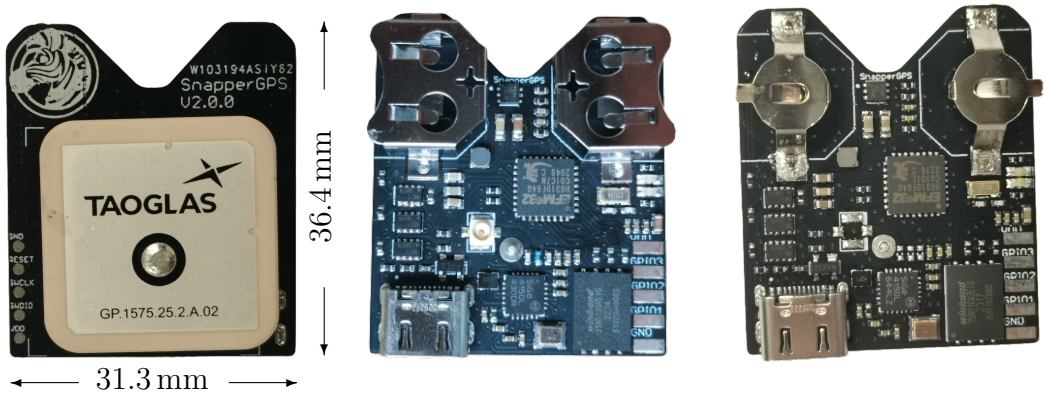


Figure 4.2: Top view (left) and bottom views (centre, right) of assembled SnapperGPS receivers V2.0.0 with a passive ceramic patch antenna and holders for two LR44 or SR44 button cells (centre) or two LR41 or SR41 button cells (right). Antenna and batteries are integrated into this design for ease of use. No components are soldered on the top of the PCB to simplify manufacturing.

the way to snapshot GNSS at low costs. Furthermore, these hardware designs are neither available open-source nor commercially. Therefore, this chapter describes a family of designs of GNSS signal snapshot loggers, members of which are shown in Figure 4.1, 4.2, 4.3, and 4.4, respectively. They target wildlife tracking in particular, an application where long battery life, low costs, low weight, small size, and long battery life are often important, cf. Section 2.14. Specifically, the following design parameters are optimised:

- Small size,

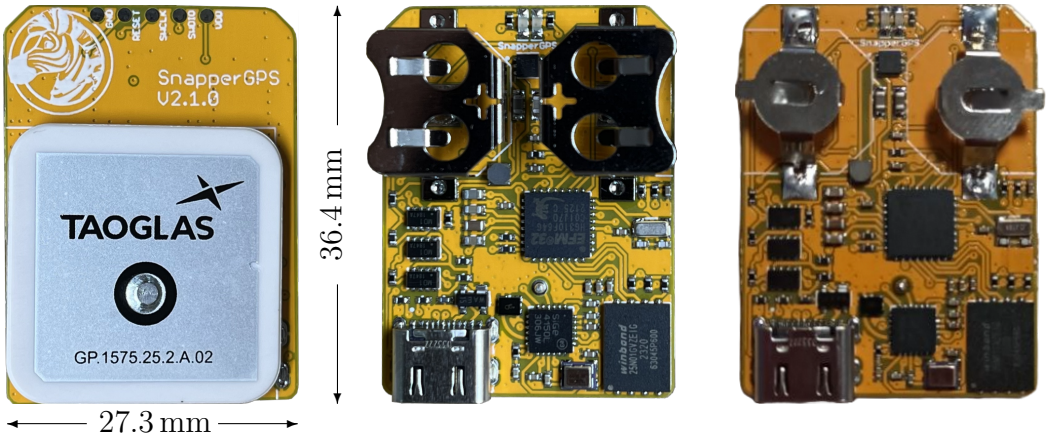


Figure 4.3: Top view (left) and bottom views (centre, right) of assembled SnapperGPS receivers V2.1.0 with a passive ceramic patch antenna and holders for two LR44 or SR44 button cells (centre) or two LR41 or SR41 button cells (right). This version is similar to V2.0.0, but does not come with exposed GPIO pins or a connector for an external antenna.

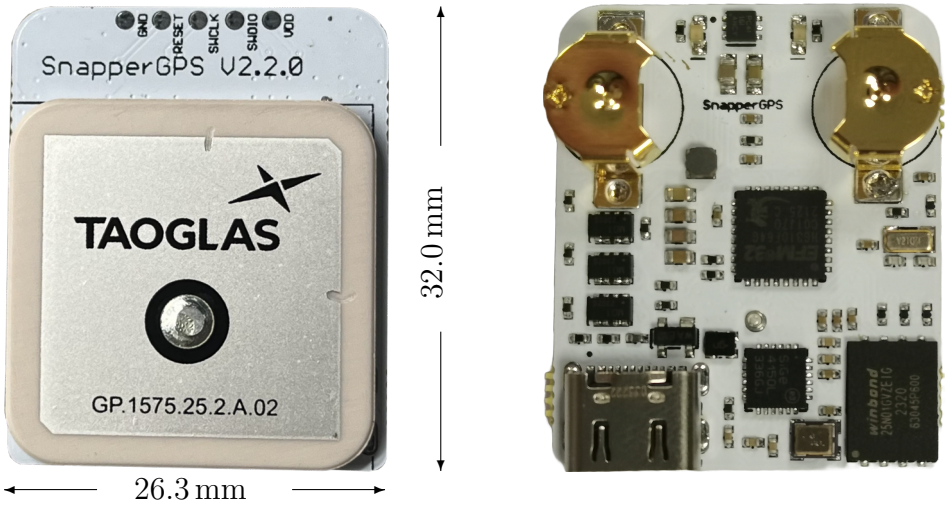
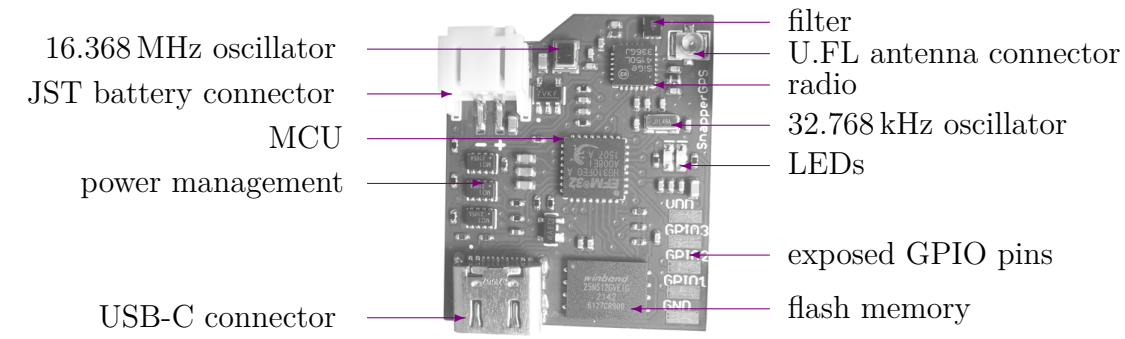


Figure 4.4: Top view (left) and bottom view (right) of assembled SnapperGPS receivers V2.2.0 with a passive ceramic patch antenna and holders for two LR48 or SR48 button cells. This version is similar to V2.1.0, but smaller.

- Low weight,
- Low energy consumption,
- Low component cost,
- Low PCB cost,
- Possibility to scale-up manufacturing,



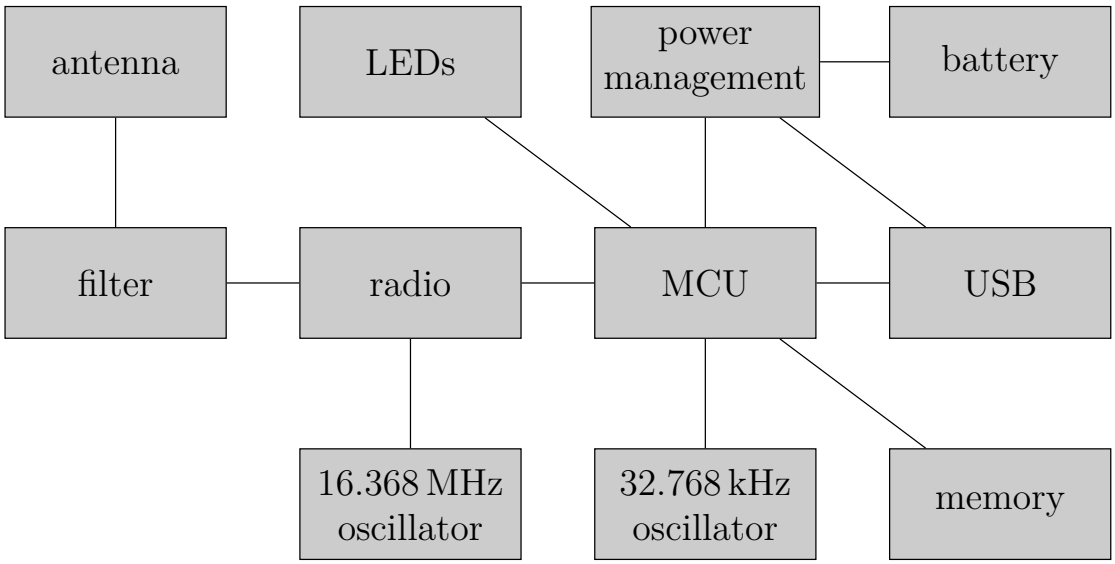


Figure 4.5: The main electronic components of a SnapperGPS receiver.

- Ease of adapting the design,
- Ease of adding a sub-module, and
- Ease of use (including by users without technical background).

The next Section 4.2 details the receiver electronics and is followed by Section 4.3 that covers the corresponding firmware. Section 4.4 briefly introduces the web application to interface with the device. Finally, Section 4.5 presents an evaluation of the system before Section 4.6 concludes this chapter.

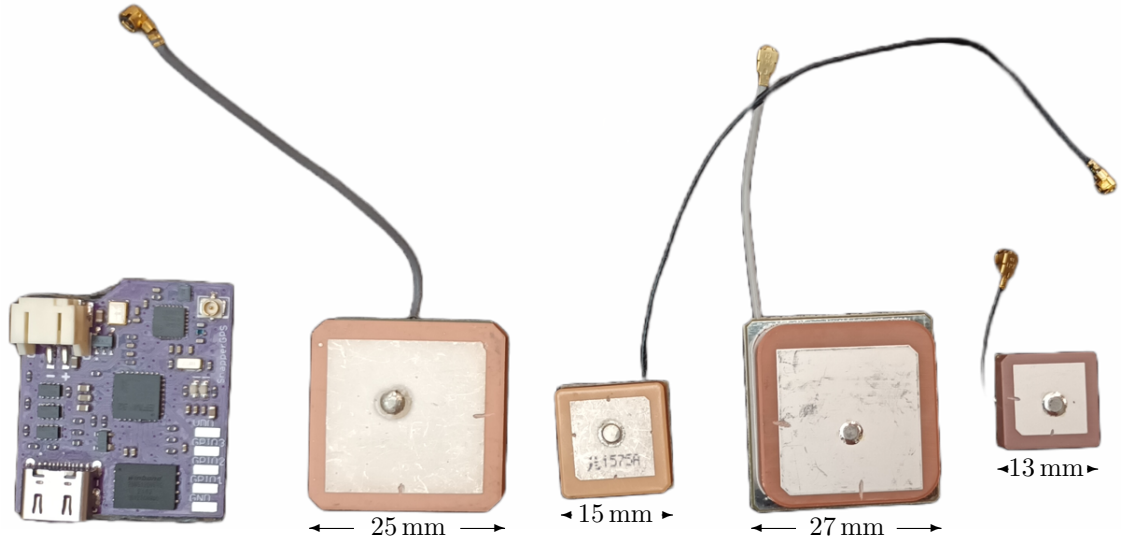


Figure 4.6: A SnapperGPS receiver and various active patch antennas it has successfully been deployed with, including a Maxtena MIA-GPS-25 (left) used in sea turtle tags, a Siretta Echo 27 (centre-left) used for creating the dataset analysed in Chapter 3 and deployed on goats, snakes, and cats, for example, an Abracon APAM2764YK0175 (centre-right) used for walking and cycling trials, and an Abracon APAM1368YB13V3.0 (right) used on sea birds. For a detailed overview of external antennas for SnapperGPS, see `https://github.com/orgs/SnapperGPS/discussions/10`. More recent deployments usually employed integrated passive antennas instead of external active antennas, see Figure 4.2.

4.2 Electronics

The block diagram in Figure 4.5 shows all core components, which are described in the following, from the left to the right following the signal flow.

4.2.1 Antenna

The antenna captures GNSS signals in the GPS L1 band, which has a centre frequency of 1.575 42 GHz. There are more GNSS signal bands, but the cheapest ICs work with the L1 band, the oldest civilian band, and they can receive the modernised GPS L1C, the Galileo E1, the BeiDou B1C, and the potential future GLONASS L1OCM signal in this band, too. This allows to make use of multiple satellite systems and, therefore, to increase localisation reliability by using more satellites.

The choice of the antenna is key for the trade-off between overall weight, costs, and localisation accuracy. Larger and more expensive antennas tend to provide a better signal-to-noise ratio (SNR), thus increasing the number of satellites that

Table 4.1: A complete SnapperGPS unit consists of a SnapperGPS receiver, an antenna, a battery, and a housing. Shown are the weights of some options. For example, a SnapperGPS board V1.0.0 with an APAM1368YB13V3.0 antenna and a 40 mA h LiPo battery weights around 8.7 g, excluding housing. A SnapperGPS board V2.1.0 with a thin on-board antenna and two LR41 batteries weights about 9.5 g.

component	mass (measured)	mass (datasheet)
SnapperGPS board V1.0.0	2.9 g	
SnapperGPS board V2.0.0 (incl. antenna)	8.4–13.7 g	
SnapperGPS board V2.1.0 (incl. antenna)	8.3–9.0 g	
SnapperGPS board V2.2.0 (incl. antenna)	8.1 g	
Echo 27 antenna	6.0 g	
APAM1368YB13V3.0 antenna	4.6 g	
Sparkfun 40 mA h LiPo battery	1.2 g	< 5 g
Sparkfun 110 mA h LiPo battery	3.0 g	< 8 g
LR44 alkaline battery (2 required)	2.0 g	1.7–2.0 g
LR41 alkaline battery (2 required)	0.6 g	0.6 g
LR48 alkaline battery (2 required)	1.0 g	0.9–1.2 g

can be identified in the signal snapshots. Another aspect is the type of antenna. GNSS antennas come in two flavours, active and passive. Active antennas have a build-in low-noise amplifier (LNA), while a passive antenna does not include an amplifier. An active antenna may sit on a separate PCB and connect to the main PCB with a coaxial cable, see Figure 4.6. Many SnapperGPS boards come with a standard U.FL connector, which allows to quickly replace the antenna and, thus, to change the size–performance trade-off, cf. Table 4.1. Most active antennas tested with SnapperGPS are ceramic patch antennas[1], which have the advantage of being compact and having a large gain towards zenith.

If there is no desire to change/customise the antenna after board assembly, then a passive antenna can be mounted on the main PCB, cf. Figure 4.2, 4.3, and 4.4. This leads to a more integrated, less fragile design and usually to lower power consumption and antenna costs. The best position for a passive patch antenna is on the centre of the back of the PCB directly underneath the subsequent components of the RF front-end to minimise the lengths of any signal transmission lines. Passive GNSS patch antennas usually have an impedance of 50 Ω, but—unlike

[1]For exemplary active antennas to use with SnapperGPS, see `https://github.com/orgs/SnapperGPS/discussions/10`.

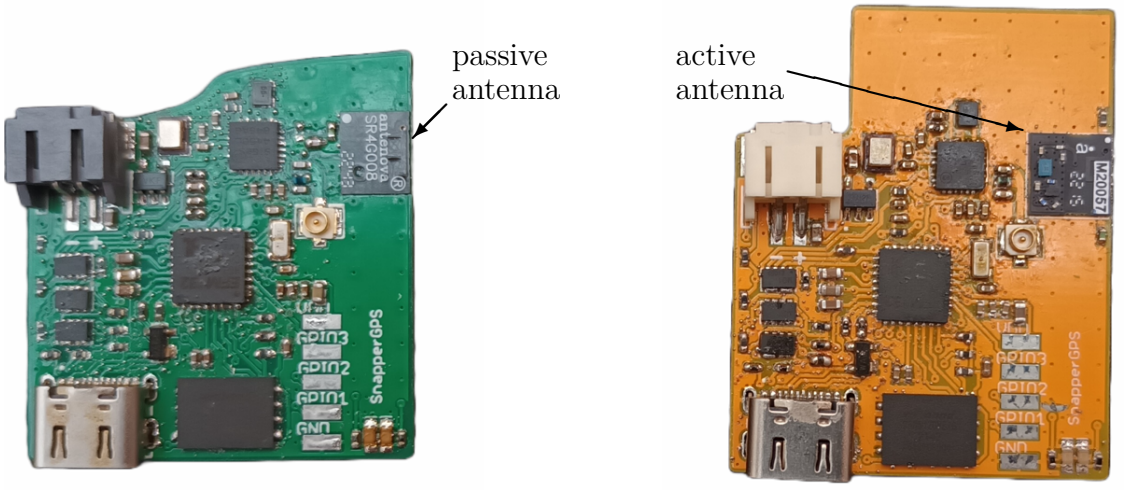


Figure 4.7: SnapperGPS prototypes with a passive PCB antenna (Antenova SR4G008, left) and an active PCB antenna (Antenova M20057-1, right), respectively.

for external active antennas—some tuning might be necessary to ensure this value when the antenna sits on a custom PCB.

Another type of antennas are PCB antennas that can consist of a trace on a small separate PCB that is soldered on the main PCB, cf. Figure 4.7. These are much lighter than ceramic patch antennas ($< 1\,\mathrm{g}$ vs. several grams). However, the received signal strength is weaker and an external matching network is usually required to bring the impedance to $50\,\Omega$. To minimise losses, output and input impedances of the components at both sides of an RF transmission line should be identical, and $50\,\Omega$ is the common choice.

4.2.2 Radio

Sampling and storing satellite signals at a rate of multiple gigahertz is not possible with a low-cost receiver. Therefore, the heterodyne method is used: by mixing the received signal with an unmodulated signal from a local oscillator, it shifts the original signal down to a lower, so-called intermediate frequency, cf. Section 2.3. For this, an integrated circuit is chosen, which is designed for low-power applications, is available at very low costs (<$1), and is designed for the GPS L1 band: the SE4150L-R from Skyworks Solutions Inc. This *radio* includes a low-noise

amplifier (LNA) for amplifying the signals of a passive antenna, a two-bit analogue-to-digital converter (ADC), and a voltage-controlled oscillator (VCO), which needs a reference frequency of 16.368 MHz from an external temperature-compensated crystal oscillator (TCXO). The SE4150L-R is designed for high-volume consumer products like navigation devices and mobile phones and hence available at low costs. Its optimal supply voltage is 3.3 V, which works for all other components, too, such that there is no need for two supply voltage domains like for Eichelberger at al. [50].

Finally, the radio has an external surface acoustic wave (SAW) band-pass filter placed next to it to suppress noise outside of the signal band.

4.2.3 Microcontroller

To sample the radio's output signal, it is connected to a low-power microcontroller unit (MCU); a 32-bit Silicon Labs Happy Gecko with an Arm Cortex-M0+ core. To reduce the overall number of components, the MCU operates on the same clock as the radio when acquiring snapshots. This allows to make use of an SPI interface with the MCU configured as master. When not acquiring snapshots, the MCU falls back to an internal 14 MHz RC oscillator.

The signal from the radio is sampled at a low 4.092 MHz rate (a quarter of the clock frequency) and with a one-bit amplitude resolution (half the radio's output quantisation), reducing the amount of captured data by a factor of eight.

4.2.4 External Flash Memory

The MCU stores the sampled signals on an external flash memory chip. Its size is the main limitation for the number of fixes. Either a 512 Mbit or a 1 Gbit flash is used, which store up to about 11,000 or 22,000 snapshots at 6 KB each, which is 12 ms worth of data.

4.2.5 Interfaces

To configure a board or download data, it must be connected to a host device. For this, a USB-C connector is added and wired to the internal USB hardware of the

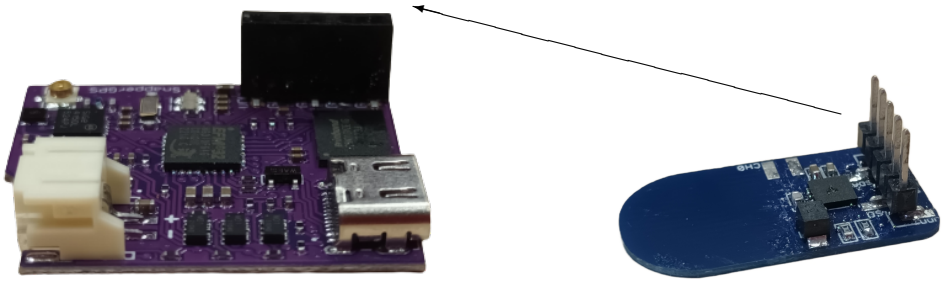


Figure 4.8: A SnapperGPS board with a standard 0.1 in header to connect a custom module (left) and such an exemplary module, a capacitive touch sensor (right).

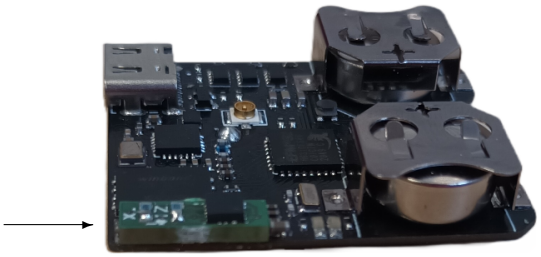


Figure 4.9: A SnapperGPS board with a custom open-source accelerometer daughter-board (front-left, green PCB) measuring 3.7 mm×11.9 mm (`https://github.com/SnapperGPS/snappergps-accelerometer-daughterboard`).

MCU. USB is chosen since almost every potential host device (including phones and tablets for configuration in the field), support the USB protocol. A USB-C connector in particular has the advantage of being small and wide-spread. Almost any owner of a recent mobile phone also owns a USB-C cable, which eliminates the need to purchase a cable for most potential SnapperGPS users.

The PCB also has two LEDs to provide information on the state of the device when it is not connected via USB.

The SnapperGPS boards expose three general-purpose input/output (GPIO) pins of the MCU as well as the supply voltage at an edge of the PCB, see Figure 4.8. These allow for the connection of external modules to extend the functionality of the device. Examples include external sensors like a barometer, a moisture sensor, or an accelerometer as well as modules for wireless data offloading. Communication can be achieved via I2C using one of the GPIO pins for the clock line and another one for the data line. Multiple options exist to physically connect a daughter-board: firstly, cables can be soldered directly on the pads of the SnapperGPS board.

Table 4.2: Comparison of different batteries used with SnapperGPS hardware. Prices from `https://digikey.com` and `https://us.rs-online.com` (button cells) and `https://www.sparkfun.com` (LiPo batteries) as of 2023.

battery	type	material	nominal voltage	capacity	weight	rechargeable	price
CR2032	button cell	lithium manganese	3 V	190–240 mA h	3–3.2 g	no	0.20–0.50 $
LR44	button cell	zinc manganese dioxide	2 × 1.5 V	120–175 mA h	2 × 1.75–2 g	no	2 × 0.20–0.60 $
SR44	button cell	silver oxide	2 × 1.55 V	150–165 mA h	2 × 2.15–2.4 g	no	2 × 1.20–1.50 $
LR41	button cell	zinc manganese dioxide	2 × 1.5 V	28–45 mA h	2 × 0.57–0.6 g	no	2 × 0.21–0.52 $
SR41	button cell	silver oxide	2 × 1.55 V	45 mA h	2 × 0.65 g	no	2 × 1.55 $
LR48	button cell	zinc manganese dioxide	2 × 1.5 V	60 mA h	2 × 0.9–1.0 g	no	2 × 0.18–0.56 $
SR48	button cell	silver oxide	2 × 1.55 V	70–77 mA h	2 × 0.9–1.2 g	no	2 × 1.29–1.90 $
DTP301120	LiPo	lithium	3.7 V	40 mA h	1.2 g	yes	4.95 $
DTP401525	LiPo	lithium	3.7 V	110 mA h	3 g	yes	5.50 $

Secondly, daughter-boards can be connected via a standard male or female 0.1 in header; see Figure 4.8 for an example. Thirdly, a small daughter-board PCB can be directly soldered on the pads; see Figure 4.9 for an example.

In addition, the reset pin and the Serial Wire Debug (SWD) pins are exposed on the back of the PCB, see Figure 4.2, 4.3, and 4.4.

4.2.6 Power Supply

The board draws its power from either the 5 V USB power line or a battery. Depending on the SnapperGPS hardware version, the battery can be a 3.7 V lithium-ion polymer (LiPo) battery (Figure 4.1), a 3 V CR2032 button battery, two 1.5 V LR44 button batteries (Figure 4.2 and 4.3), two 1.55 V SR44 button batteries, two 1.5 V LR41 button batteries (Figure 4.2 and 4.3), two 1.55 V SR41 button batteries, two 1.5 V LR48 button batteries (Figure 4.4), or two 1.55 V SR48 button batteries. They come with different disadvantages and advantages, making some of them less or more favourable for certain deployments, see Table 4.2. The high power density of LiPo batteries allows to reduce weight. Therefore, they are favourable for deployments on small animals. However, they also come with increased costs and are usually not locally available and challenging to ship internationally due to restrictions placed on them by most shippers [111]. In contrast, button cells are available in many local convenience stores at low prices, but come with higher weights. LiPo batteries also have a high self-discharge rate of about 5% per month while LR44s, for example, have a self-discharge rate of less than 10% per year [111].

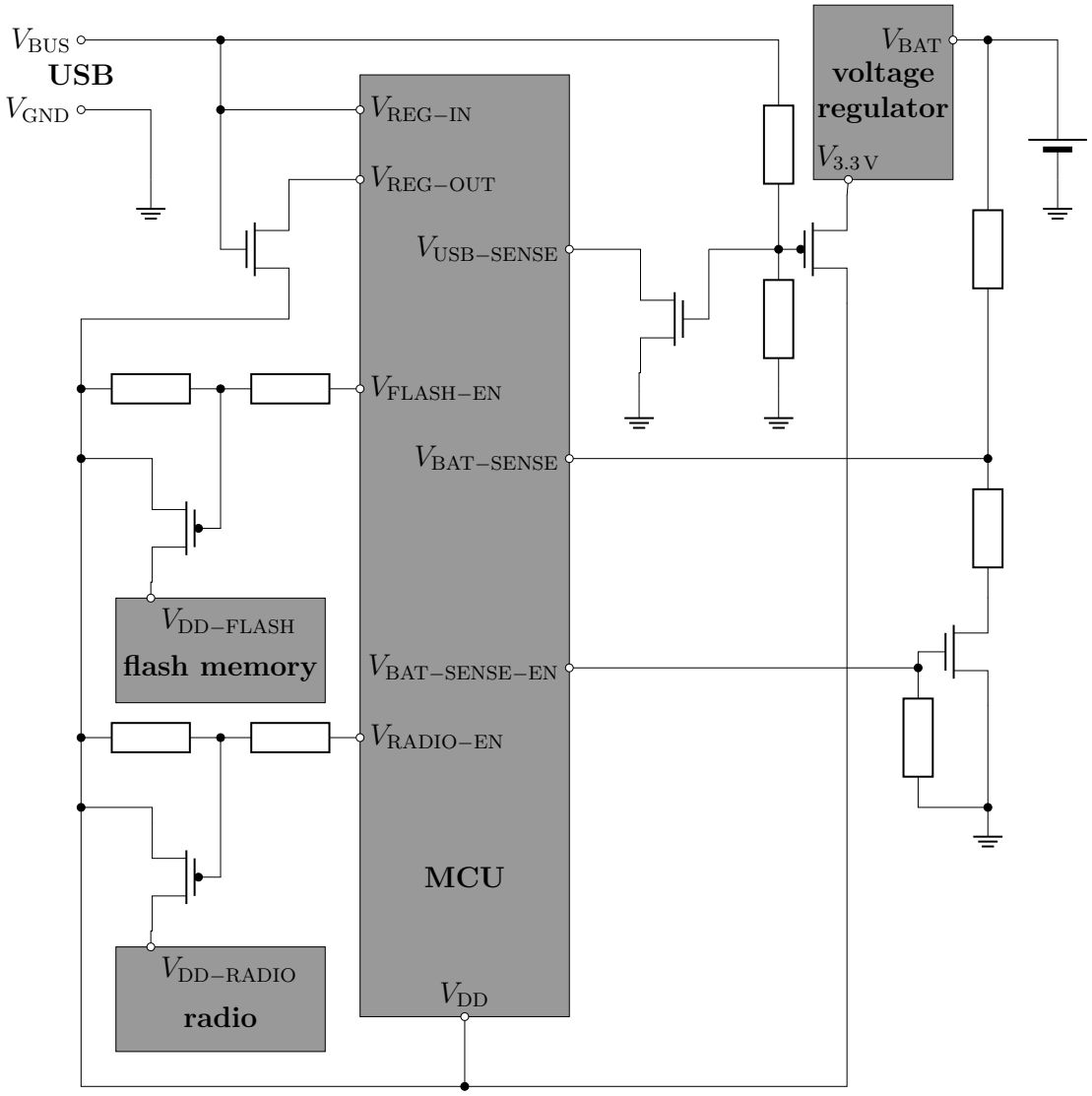


Figure 4.10: Schematic of the power circuitry of a SnapperGPS receiver V1.0.0. Power is either provided via USB (V_{BUS}, top-left) or a LiPo battery (V_{BAT}, top-right). For both sources, there is a separate voltage regulator to provide a stable 3.3 V supply voltage. The regulator for the USB supply is integrated into the MCU (V_{REG-IN} to $V_{REG-OUT}$, centre-top). The regulator for the LiPo battery is a separate IC (V_{BAT} to $V_{3.3\,V}$, top-right). A voltage divider senses whether USB is plugged in or not and switches either the USB regulator output via an N-MOSFET (top-left) or the battery voltage regulator output via a P-MOSFET (centre-top) to the supply voltage V_{DD}. Additionally, if V_{BUS} is high, then the $V_{USB-SENSE}$ input pin of the MCU is pulled low via a second N-MOSFET (top-right). Otherwise, it is pulled high by an internal pull-up resistor. A second voltage divider (right) is used to measure the battery voltage while ensuring that the measured voltage $V_{BAT-SENSE}$ is always below V_{DD}, the supply voltage of the measuring MCU. The battery measurement is turned on and off via the digital MCU output $V_{BAT-SENSE-EN}$. Finally, the two digital MCU outputs $V_{FLASH-EN}$ and $V_{RADIO-EN}$ turn the flash memory IC and the radio IC on and off utilising a P-MOSFET, respectively (left).

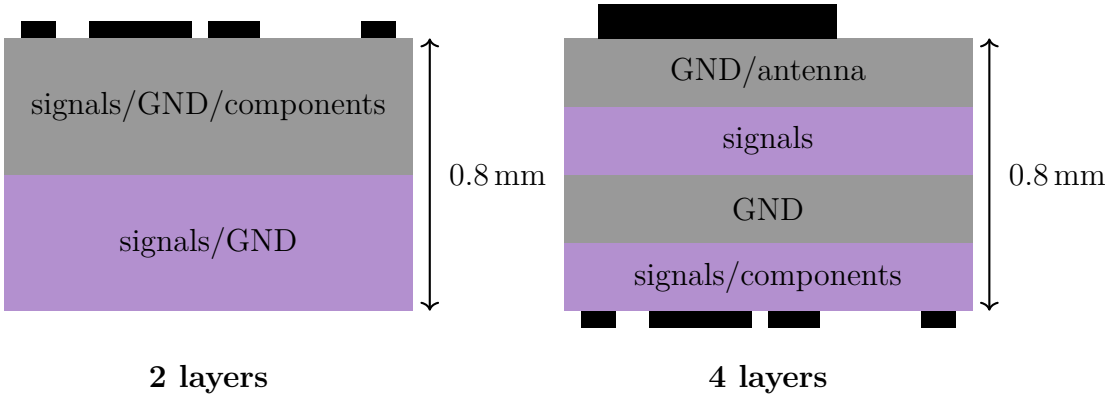


Figure 4.11: PCB stack-ups used for SnapperGPS: two-layer stack-up (left) for low-cost manufacturing and use with an external antenna; four-layer stack-up (right) to provide a continuous ground-layer underneath an integrated patch or PCB antenna on the top-layer (both diagrams not to scale).

This renders button batteries more favourable for deployments lasting many months, as long as their overall capacity is sufficient.

In each case, a power regulator and a MOSFET-based switch provide a stable 3.3 V supply voltage. However, for the button-cell-powered boards, the regulator is only used when the MCU is capturing data and by-passed when the MCU is sleeping since the current draw is small in this case and the MCU can handle lower supply voltages.

There are also MOSFETs to switch off the radio and the flash to minimise quiescent currents.

Finally, a circuit measures the battery voltage, see Figure 4.10.

4.2.7 Printed Circuit Board

All components, except for the active antenna—if present—are placed on a single PCB. Figure 4.1 shows a layout designed for low-cost manufacturing and assembly with all components placed on a single side of a two-layer PCB with signal routing on both layers. Areas between signals are filled with ground pour. The critical design part is the one with the RF signals (top-right) and is has to be taken care of the trace layout. Often, the trace width of RF transmissions lines is a critical design parameter, but this is less relevant for this design because the trace

lengths are much shorter than the GPS L1 wavelength (19 cm). However, it is important not to place digital signals close to the RF traces to avoid cross-talk. Additionally, uninterrupted copper on ground-level is underneath the RF traces to ensure low-impedance return paths.

If a passive ceramic patch antenna is integrated into the design, as in Figure 4.2, 4.3, and 4.4, then RF considerations are more relevant. Firstly, the patch antenna requires a ground plane as part of the PCB, which is why the layer next to the antenna is filled with copper on ground-level. To still have enough space for signal routing, a four-layer PCB with a ground–signals–ground–signals/components stack-up (top to bottoms) is chosen, cf. Figure 4.11. This has the disadvantage of increased PCB manufacturing costs, especially for smaller batches, though. On the other hand, the integration of all components—including the antenna—on the PCB simplifies assembly and increases robustness.

Secondly, the unamplified signals between the passive antenna and the radio are much weaker than the pre-amplified signals from an active antenna and hence need to be even more carefully protected from cross-talk from digital signals.

Digression: SnapperGPS Receivers with PCB Antennas
Passive or active patch antennas are the defaults for SnapperGPS receivers. Prototypes of SnapperGPS receivers with smaller and lighter PCB antennas were built, too, see Figure 4.7. This includes both, passive and active PCB antennas. However, in both cases the signal quality turned out to be too low to reliably achieve fixes from 12 ms snapshots in other scenarios than open sky, even when using four-layer boards with enlarged ground planes.

4.2.8 Bill of Materials

Table 4.3 presents the bill of materials (BOM) for the electronic components of an open-source SnapperGPS receiver V1.0.0 for use with a LiPo battery and an active patch antenna (Figure 4.1). The price of $21 for a single device in a batch of 100 is significantly cheaper than both, existing commercial and open-source GNSS receivers for wildlife tracking, which usually cost a few hundred USD, cf. Section 2.14.

Table 4.3: Bill of materials for one, 100, and 1000 SnapperGPS receivers V1.0.0 (prices as of 2022). For a continously updated table, see `https://github.com/SnapperGPS/snappergps-pcb/blob/main/README.md`. For the bill of materials of V2.0.0, V2.1.0, and V2.2.0, see `https://github.com/SnapperGPS/snappergps-pcb-2/blob/main/README.md`, `https://github.com/SnapperGPS/snappergps-pcb-2-1/blob/main/README.md`, and `https://github.com/SnapperGPS/snappergps-pcb-2-2/blob/main/README.md`, respectively.

Component	Value	Part	Qty	Price [USD]		
				one	100	1000
Ceramic capacitor	100 nF	GRM155R60J104KA01D	10	0.33	14.60	81.70
Ceramic capacitor	10 nF	GRM155R60J103KA01D	5	0.50	4.43	29.10
Ceramic capacitor	100 pF	GRM1555C1H101JA01J	1	0.10	0.81	4.54
Ceramic capacitor	22 pF	GRM1555C1H220JA01D	1	0.10	0.81	4.57
Ceramic capacitor	18 pF	GRM1555C1H180JA01D	2	0.20	2.32	13.00
Multi-layer ceramic capacitor	1 uF	GRM188R60J105KA01D	4	0.40	11.00	61.84
Multi-layer ceramic capacitor	4.7 uF	GRM185R60J475ME15D	1	0.30	11.67	72.02
Multi-layer ceramic capacitor	10 uF	GRM188R60J106KE47D	1	0.15	5.07	29.44
Fixed inductor	39 nH	0402HS-390EKTS	1	0.26	15.68	109.72
Ferrite bead		BLM15HB221SH1D	1	0.10	4.36	25.08
Resistor	15 Ω	RC0402FR-0715RL	2	0.20	1.32	5.98
Resistor	1 kΩ	RC0402FR-071KL	2	0.20	1.32	5.98
Resistor	4.7 kΩ	RC0402FR-074K7L	3	0.30	1.98	7.77
Resistor	5.1 kΩ	RC0402FR-075K1L	2	0.20	1.32	5.98
Resistor	6.8 kΩ	RC0402FR-076K8L	1	0.10	0.66	2.99
Resistor	10 kΩ	RC0402FR-0710KL	2	0.20	1.32	5.98
Resistor	100 kΩ	RC0402FR-07100KL	7	0.70	4.62	14.98
Crystal	32.768 kHz	Q 0,032768-JTX310-12,5-10-T1-HMR-50K-LF	1	0.86	62.70	495.00
Yellow-green LED	572 nm	SML-D12M1WT86	1	0.22	5.51	39.86
Red LED	620 nm	SML-D12U1WT86	1	0.22	5.51	39.86
Temp. compensated oscillator	16.368 MHz	D32G-016.368M	1	8.00	700.00	7000.00
GPS receiver		SE4150L-R	1	0.86	86.40	864.00
SAW filter	1.57542 GHz	AFS20A42-1575.42-T3	1	1.05	71.37	543.74
Microcontroller		EFM32HG310F64G-C-QFN32R	1	5.91	435.04	3703.46
TVS diode	5.5 V	PRTR5V0U2AX,235	1	0.66	42.47	257.87
NAND flash memory	512 Mbit	W25N512GVEIG	1	2.90	228.68	2051.92
U.FL connector jack	50 Ω	1909763-1	1	0.51	36.55	595.90
MOSFET array N/P-channel		TT8M1TR	3	1.56	86.58	487.08
Linear voltage regulator		S-1318D33-M5T1U4	1	1.58	109.94	908.20
USB-C receptacle connector		DX07S016JA1R1500	1	1.67	129.65	1080.44
JST connector header		S2B-PH-SM4-TB	1	0.54	39.08	339.84
Total				31.55	2124.54	18900.69

The BOM differs slightly from the one for initial SnapperGPS test boards, which was only $14. For example, a cheaper TCXOs was used, the X1G005441030112 manufactured by EPSON or the 7Q-16.368MBG-T from the TXC Corporation, which cost around $2, instead of the expensive D32G-016.368M, which costs $7–$8. This change had to be introduced because of the on-going global supply chain disruptions and it is recommended to revert to one of the initial choices for replications, if possible, to reduce the overall cost.

To replicate SnapperGPS, it is possible to send the BOM and the provided Gerber, drill, and assembly files to any commercial PCB assembly company, which in turn will source the components, print the PCB, and assemble the board. An

alternative is to obtain just SnapperGPS PCBs from a manufacturer and to manually assembly the boards. Hand-soldering a board takes less than 2 h for a skilled person, but requires intermediate equipment and skills, see the documentation in the snappergps-pcb repository for details[2].

	resistivity [Ω m]	relative permittivity	relative permeability
air	10^9–10^{15}	1.00	1.0000004
sea water	0.2	72–81	0.999991
fresh water	20–200		

Table 4.4: Comparison of resistivity, relative permittivity, and relative permeability of air, sea water, and fresh water at a temperature of 20 °C [112–114].

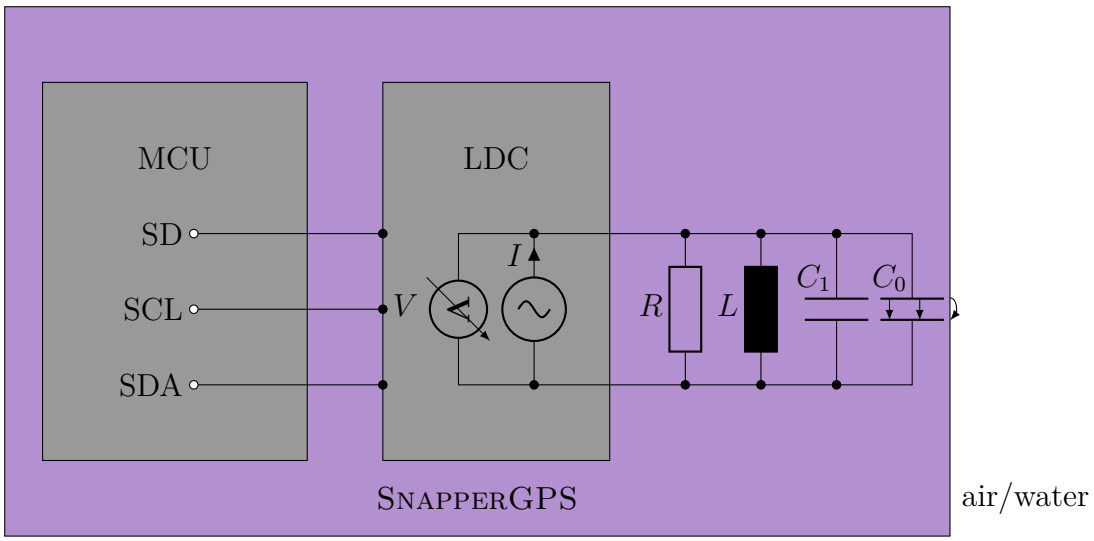


Figure 4.12: Surfacing detection circuit based on the different permittivity of air and water using an inductance-to-digital converter (LDC).

Digression: Surfacing Detection for Tracking Aquatic Animals with Snapshot GNSS

Using snapshot GNSS to track aquatic animals (cf. Section 2.15) poses the question when to trigger a snapshot capture to avoid filling the snapshot receiver's internal storage with data that was collected underwater and hence does not contain any satellite signals [90].

To detect whether a tag is inside or outside of water, an electronic circuit can

[2]Hardware design files, assembly and testing instructions, and more can be found on `https://github.com/SnapperGPS/snappergps-pcb`, `https://github.com/SnapperGPS/snappergps-pcb-2`, `https://github.com/SnapperGPS/snappergps-pcb-2-1`, and `https://github.com/SnapperGPS/snappergps-pcb-2-2`, respectively.

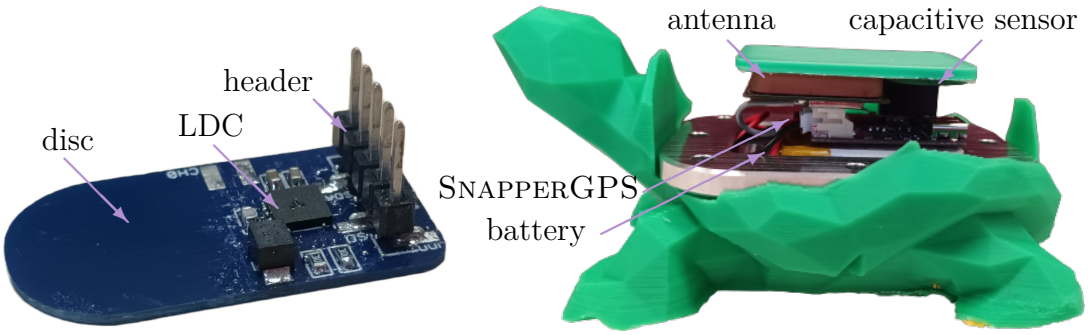


Figure 4.13: Left: Capacitive-sensing daughter-board for a standard SnapperGPS board. The daughter-board is a two-layer PCB. The capacitor is realised as a disc on a single layer of the PCB and is on the left, the LDC is in the centre, and the header to connect the SnapperGPS board is on the right. **Right:** Mock-up with the interior of a sea turtle tag with LiPo battery (bottom), SnapperGPS board (centre), active patch antenna (top-left), and the capacitive sensing daughter-board (top-right).

exploit the different electrical properties of air on the one hand and (salt) water on the other hand. Their conductivity, permittivity, and permeability differs, see Table 4.4, such that an electronic circuit where the medium surrounding the tag (i) forms a resistor, (ii) is the dielectric of a capacitor, or (iii) is penetrated by the magnetic field of an inductor can be used. The change in resistance, capacitance, or inductance can be monitored to determine whether the tag is in air or water, respectively. SnapperGPS was deployed with a system based on (i), a system based on a combination of (i) and (ii), and a system purely based on (ii). A system measuring a difference in permeability (iii) was prototyped, too, but not employed in practise because of the small difference in permeability between air and water and the resulting high sensitivity to parasitic effects.

For the first system, two electrodes are realised as screws outside of the tag. In regular intervals, the MCU leaves its low-power sleep mode and indirectly measures the resistance between the two electrodes. It determines the time it takes to charge a capacitor through this resistor formed by the sea water or air between the electrodes to a certain voltage level.

The second system also uses the two electrodes outside of the tag. This time, the water/air between the electrodes is modelled as an RC circuit itself, not just as a resistor. In regular intervals, the capacitor formed by the two electrodes is charged to a certain voltage. Then, the time is measured until its voltage drops below a threshold due to the capacitor discharging via the resistance between the electrodes. In air, capacitance and resistance are high (cf. Table 4.4), and discharging takes a long time. With a water film between the electrodes, the capacitance is still mainly determined by the properties of air, while the resistance mainly depends on the small resistivity of water. In consequence, the discharging is much faster. Finally, a fully submerged tag has a low resistance and a higher capacitance than in air.

Figure 4.14: Top-left: Bottom view of a SnapperGPS board with integrated copper disc and LDC at the top. **Top-right:** Top view of the same board with an integrated passive patch antenna. **Bottom-left:** 3D model of the board in a custom enclosure. **Bottom-right:** Custom CNC-milled enclosures made of POM with an o-ring for sealing. The enclosures have a low profile to minimise drag and are pressure tested to be waterproof at 12 bar, which corresponds to about 120 m water depth.

Therefore, the discharge rate is similar to that of the second case, but the overall discharge time is longer due to the higher initial charge. A potential advantage of this design over the purely resistance-based one is that it allows to distinguish a thin water film that can be penetrated by GNSS signals from a fully submerged tag that cannot receive GNSS signals.
In addition to the potential problem of the purely resistance-based method w.r.t. a water film on the tag surface, both methods described above face the issue that the two electrodes are exposed outside of the tag. This (i) can cause them to change their electrical properties during a weeks-long deployment in an ocean, for example, due to fouling [90], water pooling, abrasion, or sedimentation of particles and (ii) poses the thread of leakage. Therefore, the third system employs a contact-less sensing method based on capacitance measurements only. The capacitive sensing circuit (Figure 4.12) is entirely placed on either a daughter-board (Figure 4.13) or a separate section of a SnapperGPS board (Figure 4.14). Its core element is a capacitor C_0 realised as a copper disc on the top layer of the PCB. The electrical field of the disc capacitor penetrates the space surrounding the tag. In consequence, if the permittivity of the material surrounding the tag changes, the capacitance of the disc capacitor changes, too. A disc is chosen as shape to minimise stray fields and focus the field right above the disc itself [115]. In addition to the disc capacitor, the sensor consists of a capacitor C_1 and a coil L realised as discrete surface-mounted components, respectively, and an inductance-to-digital converter (LDC) IC in parallel. The LDC can apply an AC current with constant amplitude I to the circuit and measure the resulting voltage amplitude V. The coil L and the capacitors C_0 and C_1 form together with the parasitic resistance R of the circuit an R-L-C resonator. At a constant drive current amplitude I and sweeping input frequency f, the impedances of L and $C = C_0 + C_1$ cancel at the resonance frequency f_0 and the voltage amplitude V is maximised. The resonance frequency is approximately $f_0 \approx \frac{1}{2\pi\sqrt{LC}}$. Since L is approximately constant, only a change in C affects $f_0 \propto \frac{1}{\sqrt{C}} = \frac{1}{\sqrt{C_0+C_1}}$. Since the PCB with the copper disc is mounted directly underneath the top of the tag enclosure, a change of the permittivity ϵ of the material surrounding the tag (water or air) affects the capacitance of the disc $C_0 \propto \epsilon$ and, hence, f_0. To constrain the magnitude of the capacitance change and keep it in the range of the ADC of the LDC, a discrete surface-mounted capacitor C_1 is added in parallel to the disc.

4.3 Firmware

The Happy Gecko MCU runs custom firmware written in the C programming language. The state machine in Figure 4.15 visualises its core functionalities. To minimise the overall energy consumption, the firmware carefully selects for each

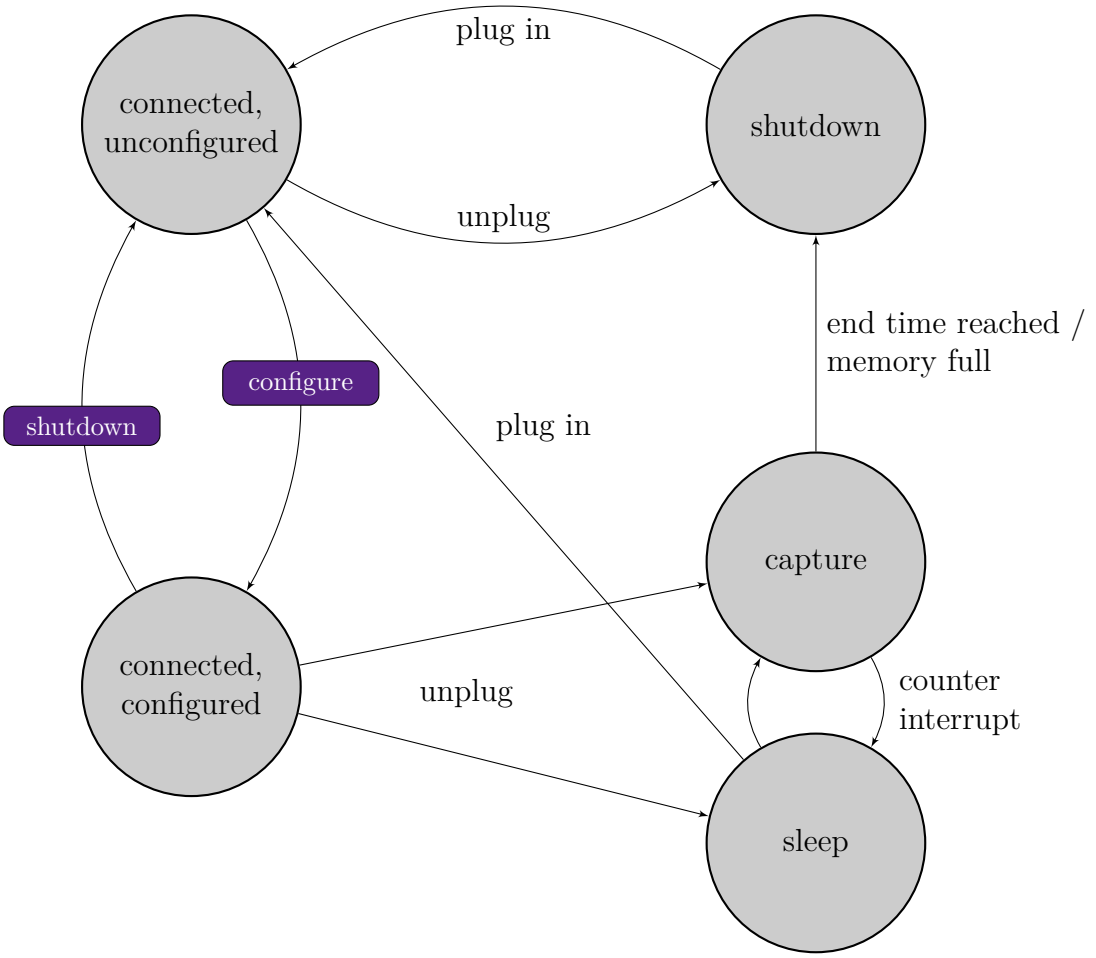


Figure 4.15: State machine of a SnapperGPS receiver.

state one of the five energy modes (EMs) that the Happy Gecko offers [116]. Almost all the time during a deployment, the MCU sleeps in stop mode (EM3), which consumes only 0.5 µA. A real-time counter (RTC) triggers wake-ups in regular intervals to record a satellite signal snapshot, respectively, and the MCU switches to run mode (EM0) for this. The flow chart in Figure 4.16 shows the steps that it completes in this mode before it goes to sleep again.

The radio outputs satellite signal samples at 16.368 MHz with two-bit amplitude quantisation. However, the MCU samples the signal at a low 4.092 MHz rate, a quarter of the clock frequency. Additionally, it uses only one bit for the amplitude quantisation, the sign and not the magnitude. Both together reduce the amount of acquired data and, hence, required memory by a factor of eight. However, the main advantage is that the existing device USART configured as an SPI master can capture

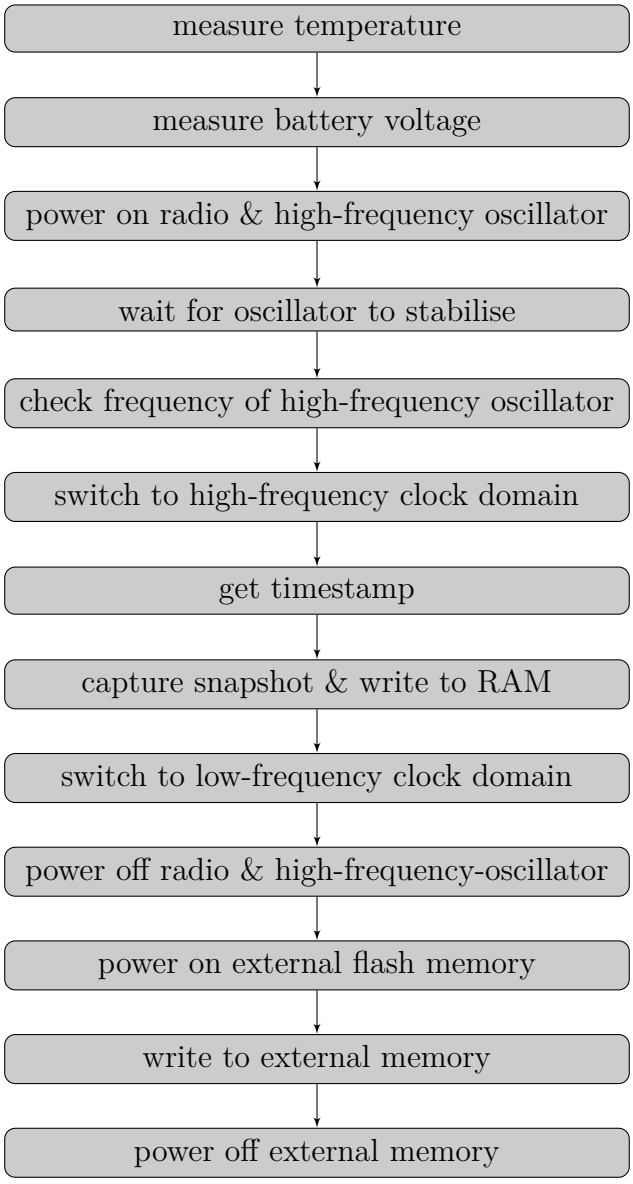


Figure 4.16: Flow chart for capturing a snapshot.

the one-bit receiver output. In contrast, the solution of Eichelberger et al. demands for two bits being sampled at 16 MHz, which limits their MCU choice to one with a parallel input capture interface (PARC) [50], while Liu et al. use additional circuitry to convert their two-bit GPS input stream at 16 MHz into a 16-bit parallel signal at 2 MHz, which an MCU then captures using direct memory access (DMA) [32].

The choice for SnapperGPS gives 32 clock cycles to acquire an eight-bit sample and write it to RAM. The small RAM size of the low-cost MCU limits the snapshot length to 6 KB, i.e., 12 ms. Afterwards, the MCU switches off the radio and the

high-frequency oscillator and writes the snapshot together with a timestamp and measurements of the battery voltage and the temperature via SPI to the external flash memory, which it only powers on for this purpose.

Subsequently, the MCU goes to sleep in EM3 again or switches to shutoff mode (EM4) if all snapshots have been captured and the deployment is complete. In shutoff mode, the RTC is also powered down and the MCU consumes just 20 nA.

To preserve energy, the MCU switches off the high-frequency 16.368 MHz oscillator during sleeping because it only needs a high frequency clock while capturing a snapshot and the low-frequency 32.768 kHz oscillator is sufficient to keep track of time between consecutive snapshots and the RTC running.

A GPIO pin is configured with a pull-up resistor to be pulled low when a USB connector is plugged in. Then, the MCU leaves EM3 or EM4, stops the RTC and, therefore, the recording of further snapshots, and switches to EM0. A SnapperGPS receiver provides USB bulk endpoints in both directions to facilitate fast transfers of larger amounts of data. These enable it to be configured as a WebUSB device, for example, together with the SnapperGPS web application. The web application is linked in the USB descriptor of the device, allowing the operating system of the host device to direct the user to the homepage. No driver installation or app download is necessary on almost[3] every common mobile or desktop operating system.

The documentation in the snappergps-firmware repository[4] defines the messages to communicate with a SnapperGPS receiver via USB. The messages can be used to get status information from the receiver, configure a receiver for a deployment, including setting start and end time and the time interval for snapshot capture, and transfer snapshots and timestamps to a host computer.

The latter requires transferring snapshots from the external flash memory to the MCU RAM first and then via USB to the host, one snapshot at a time.

In addition, USB is used to send a firmware update via the MCU to the external flash memory and to request a cyclic redundancy check (CRC) afterwards to detect

[3]Only iOS and iPadOS do not support WebUSB among the major mobile or desktop operating system as of 2023.

[4]`https://github.com/SnapperGPS/snappergps-firmware`

transmission errors. Next, a custom bootloader function that is held in SRAM copies the firmware update from the external flash memory to the flash memory of the MCU before the MCU reboots to complete the update.

The firmware comes as a compiled binary with a size of about 29 KB and can be compiled with the Arm GNU Toolchain. It can be flashed to an assembled SnapperGPS receiver via USB.

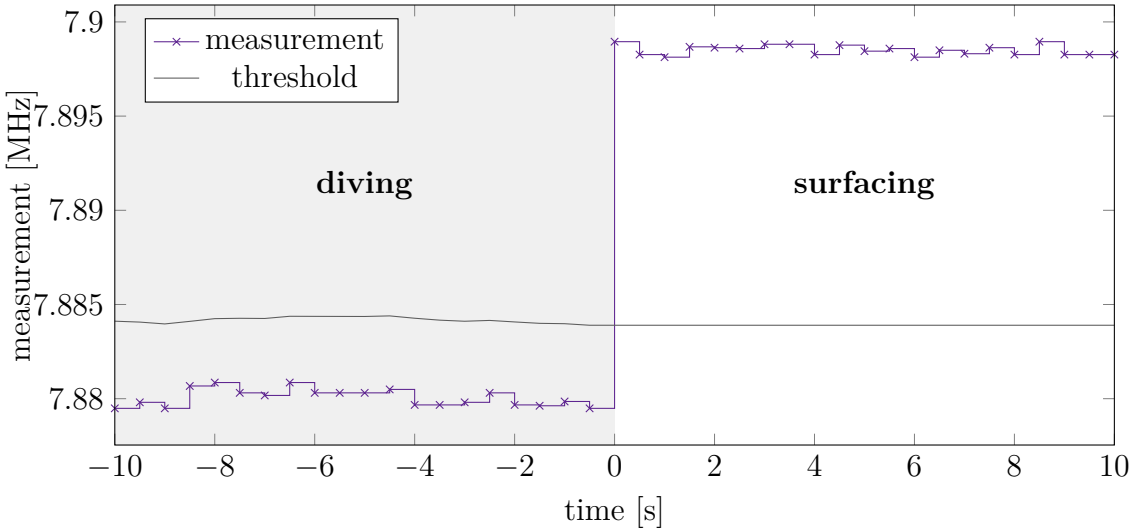


Figure 4.17: Exemplary signal from a capacitance-based surfacing-detection sensor when removing the tag from a bowl of salt water. The sampling frequency is 2 Hz.

Digression: Snapshot Acquisition Scheduling for Aquatic Animals
To make use of the optional electronics for surfacing detection described in the previous box, the standard firmware needs to be modified to accomplish two additional tasks:

1. Read out the circuit's measurement and estimate whether the tag is below or above the surface, and
2. Based on this estimate and an overall strategy, decide whether to acquire a satellite signal snapshot or not.

Figure 4.17 shows an exemplary measurement sequence of the circuit that aims to detect surfacing based on a change of capacitance only. The probability distribution of the LDC measurements is modelled as a Gaussian mixture with two Gaussian distributions, one for diving and one for surfacing. The distributions' means are fitted using an exponentially weighted moving average to account for drift. Surfacing events are then triggered when the measurement overshoots a three-sigma confidence value above the diving distribution's

estimated mean and diving is recognised by falling back into the distribution's confidence bounds.
During a deployment, the number of collected snapshots should roughly increase linearly over time. For this, a decision rule is used such that a tag collects data when it is assumed to be above the surface, but never more snapshots than a linear acquisition strategy would allow. If estimated to be under water, the tag does not collect data, until there are by a certain margin less data points than expected from a linear acquisition strategy. Then, it starts collecting data, too. This is to account for a malfunctioning tag that detects no or few surfacing events. In addition, heuristics are added for robustness. For example, if a surfacing phase lasts longer than a time threshold, then a missed diving event is assumed.

4.4 Web Application

The previously introduced SnapperGPS web application[5] (Figure 4.18, Section 3.4), exposes three core interactive views to the user.

4.4.1 Interface between Web Application and Receiver

One of these views is the *configuration view* to connect a SnapperGPS receiver via WebUSB and configure it for a subsequent deployment, read out information such as the battery voltage, set parameters such as the on-board clock time and the recording interval between two consecutive snapshots, and start a recording. It can

[5] `https://snappergps.info`

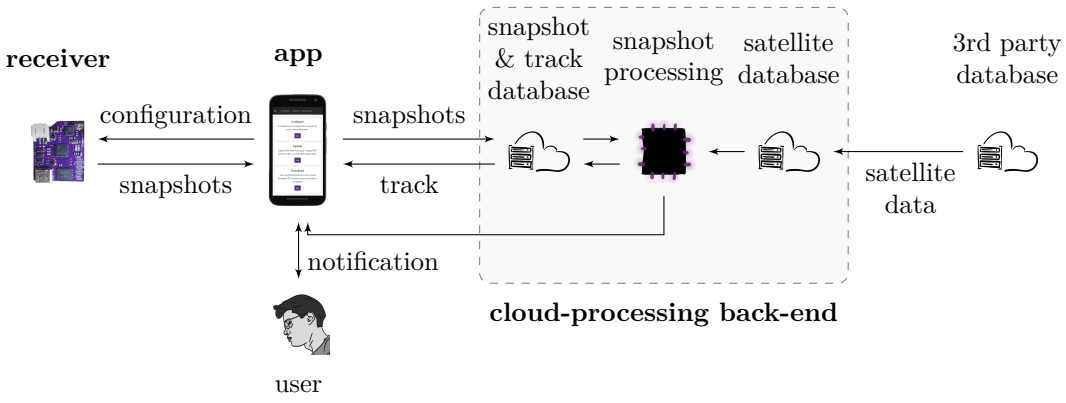


Figure 4.18: The SnapperGPS system comprising receiver, web application, and cloud-processing back-end.

also be used to update the firmware via USB (see the firmware description presented in Section 4.3). Since the SnapperGPS app communicates with a SnapperGPS board via WebUSB, no driver or other software installation is required.

The application is a progressive web app (PWA), which means that it can run in a browser window, but also as a native app. Additionally, PWAs load quickly, require little memory, are secure to use, and a service worker automatically keeps the app up-to-date. The SnapperGPS app is compatible with the Android mobile operating system and runs offline. Therefore, it can be used to configure and read out SnapperGPS receivers in the field. The SnapperGPS app is tested in the Microsoft Edge and Google Chrome browsers. Optionally, the user can also install it as a native app, either through the website[6], the Microsoft Store[7], or Google Play[8].

After a deployment, an upload view transfers snapshots and timestamps from the device to a server. Optionally, the user can request to be notified about the completion of the processing of the their data via a push notification, an email, or a Telegram message[9].

4.4.2 Automatic Calibration

A SnapperGPS receiver does not require manual calibration despite the fact that it is made of low-cost components with relatively wide tolerances. This is because the software automatically estimates and corrects errors.

The main components of concern are the two oscillators. The SnapperGPS receiver uses the 32.768 kHz oscillator to time stamp snapshots. The snapshot GNSS method requires timestamps with an accuracy of at least 1 min [19]. However, the 32.768 kHz oscillators that SnapperGPS receivers use have a frequency tolerance of at least ±10 ppm, which can cause an error of over 5 min during a year-long deployment. The snapshot GNSS back-end mitigates this by estimating the time error for every single snapshot and using the previous estimate as initialisation

[6] `https://snappergps.info`
[7] `https://apps.microsoft.com/store/detail/snappergps/9P9RPRS6LSMM`
[8] `https://play.google.com/store/apps/details?id=com.herokuapp.snapper_gps.twa`
[9] `https://t.me/SnapperGPSBot`

for the next one. Since the on-board time is synchronised with the browser time during configuration, this propagation is sufficient to keep track of the time error during the location estimation process.

The synthesizer of the radio turns the 16.368 MHz reference frequency of the external TCXO into a frequency close to the GPS L1 frequency of 1575.42 MHz. Therefore, frequency errors of the TCXO are amplified by almost a factor 100 and a frequency tolerance of ±500 ppb can turn into a frequency offset of the front-end of more than 800 Hz, which shifts the intermediate frequency of the captured snapshots by the same amount. However, for successful satellite acquisition, an accuracy that is a magnitude better than that is needed, especially for Galileo and BeiDou satellites. To avoid searching a large frequency space for the satellite signals, the back-end automatically estimates the offset of the receiver front-end for each uploaded dataset using a couple of snapshots from the beginning of the deployment, cf. Section 3.2.1. Afterwards, it corrects the offset estimate over time based on the recorded temperatures and a linear model that relates temperature and frequency offset and which was fitted to a training dataset. This allows to only search a band of ±200 Hz around the centre frequency predicted for each satellite based on this offset and its Doppler shift.

Both steps together allow the SnapperGPS system to obtain reliable localisation accuracy without manual calibration.

4.5 Evaluation

This chapter's evaluation of the SnapperGPS hardware considers localisation accuracy (Section 4.5.1) and energy consumption (Section 4.5.2). Evaluations in real-world wildlife tracking studies can be found in Chapter 5 together with results from a user survey.

4.5.1 Accuracy

Chapter 3 already showed median tracking errors of 11–12 m on an open test data collection that was established using three SnapperGPS receivers. It consists

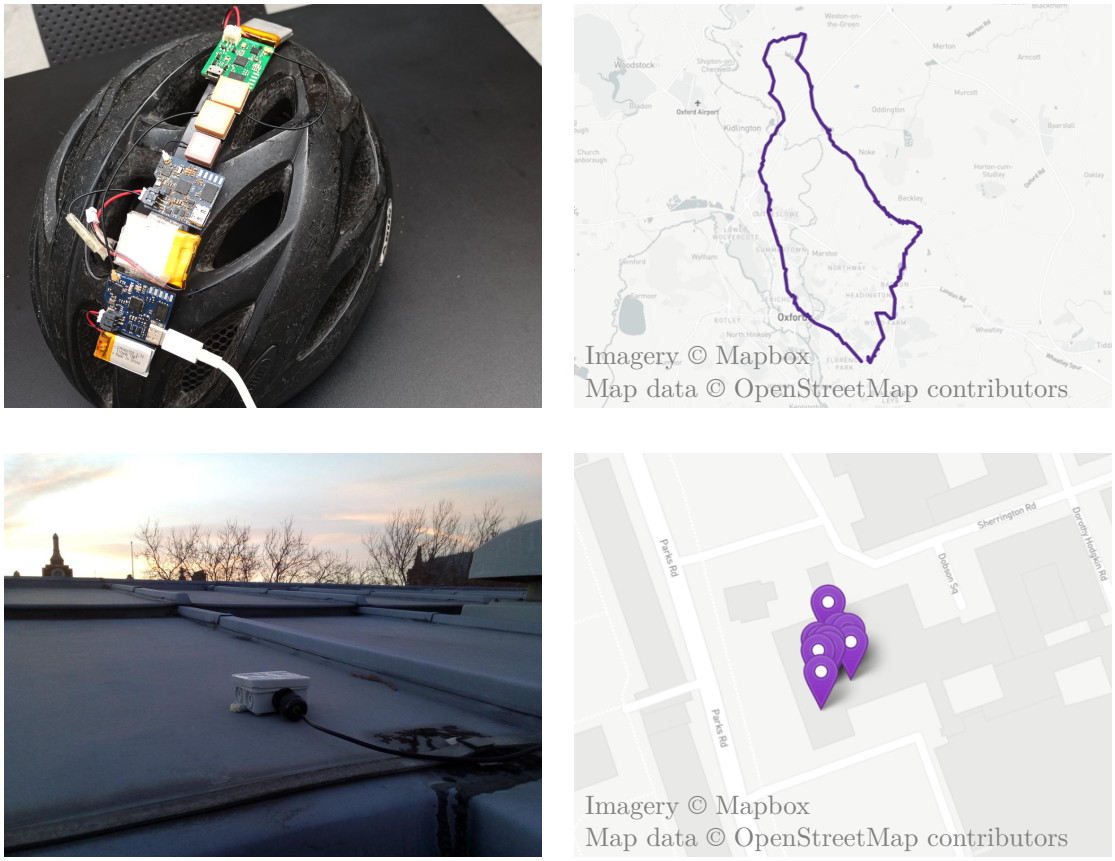


Figure 4.19: Three SnapperGPS receivers on a cycling helmet (top left), example track captured while cycling and walking in and around Oxford, UK (top right, from `https://snappergps.info/view?uploadid=8821aa36c3`), roof-mounted SnapperGPS receiver for stationary data collection over twenty-one months (bottom left), and example set of ten fixes (bottom right). More exemplary tracks can be found on `https://snappergps.info/view`.

of four static and seven dynamic cycling tests in urban and rural environments with 3700 snapshots in total. Additionally, a receiver mounted on a roof top with good sky visibility collected about 23000 snapshots over more than 22 months and achieved a median error of 9.7 m (see Figure 4.19).

4.5.2 Energy Consumption

Measuring the power/current consumption of a SnapperGPS receiver is critical for long-term deployments to ensure that it will work for the desired operation time. The current is 1–2 µA if the board is sleeping and the maximum current should be around 25 mA when capturing a snapshot. The charge consumption for a single snapshot is <0.3 µA h.

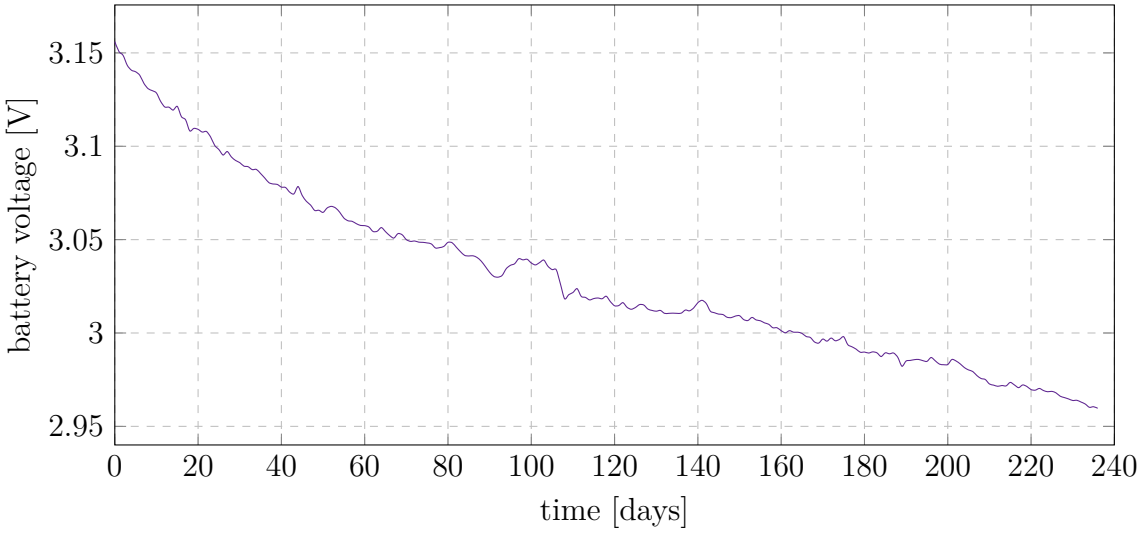


Figure 4.20: Exemplary voltage profile of a SnapperGPS receiver with an active antenna powered by two 158 mA h LR44 batteries (brand: RS Pro) capturing a GNSS signal snapshot every 15 min over the course of about eight months. The batteries have a nominal voltage of $2 \times 1.5\,\mathrm{V} = 3\,\mathrm{V}$. The drop of the voltage from 3.16 V to 2.96 V corresponds to a change of the state-of-charge from 99% to 93% or by 9 mA h [117].

The current consumption is an improvement over the solution of Eichelberger et al. [50], which achieves only a higher current of 2.9–5.1 µA on average during stand-by with more expensive hardware.

The two values reported above allow to calculate deployment durations for a given battery. For example, a 15 mA h battery could operate a board for a year. However, the battery self-discharge needs to be accounted for on top of this. LiPo batteries self-discharge by roughly 5% per month (1-2% in the battery itself plus 3% because of the safety circuit), although, the exact value depends on the quality of the battery, its protection circuit, temperature, humidity etc. [118]. Button cells usually self-discharge less than 10% per year [111]. Nevertheless, even considering the self-discharge, a SnapperGPS receiver can operate for more than a year on either a 40 mA h LiPo battery or two 28–45 mA h LR41 button cells. Both configurations weight about 1.2 g.

Figure 4.20 shows the discharge curve of two larger LR44 button cells during operation of a SnapperGPS receiver.

Digression: On-Board Satellite Acquisition
The bottleneck that prohibits using a low-bandwidth wireless connection—such as a low-power wide-area network (LPWAN) or a satellite up-link—for cloud-offloading of data from a pure snapshot GNSS receiver is the large size of an individual snapshot of several kilobytes, 6 KB in the case of SnapperGPS. The most straight forward way to eliminate the need for transmitting full snapshots is to run the satellite acquisition stage (Section 2.4, Section 3.2.1) on the device, i.e., to calculate code phases and SNRs on board. This would reduce the amount of data to transmit to two values per satellites.
However, if no satellite navigation data is available on the device, the receiver would need to spend a lot of compute to search a large frequency space for a large number of satellites. This is because no prior knowledge is available on whether a satellite can be expected to be visible or not and approximate Doppler shifts are also unknown.
So, the availability of some assistance data, i.e., satellite navigation data or an almanac, would be beneficial. This could either be achieved by pre-loading this data for deployments with durations of up to a month or through the availability of a down-link with a capacity of at least 10 KB per day.
Acquisition of Galileo's E1 signal and especially GPS' L1C and Beidou's B1C signals still remains a computationally heavy task due to their longer codes and codes with multiple components, cf. Section 2.2. Therefore, an option is to run on-board satellite acquisition for GPS L1 signals only and to refine the locations after tag recovery using all GNSS.
However, implementing the signal down-mixing and FFTs needed for satellite acquisition (Section 2.4) on a low-power MCU is still challenging and basically impossible on the Cortex-M0+ core of the EFM32 Happy Gecko used for SnapperGPS. A proof-of-concept implementation on an EFM32 Wonder Gecko with a Cortex-M4 core still takes several minutes for a single satellite. Since the operations could well be parallised, including the use of a single channel for each satellite, a dedicated digital signal processor (DSP) should be used for this task—or a field-programmable gate array (FPGA), which usually offers hardware support for FFTs.

4.6 Conclusions

Together with the novel algorithms from Chapter 3, the purpose-built SnapperGPS snapshot receivers form a snapshot-positioning system with an unmatched trade-off between costs (<$30 receiver), energy consumption (15 mA h/year at 3.3 V, less than the energy in 8 mg of chocolate), and performance, which is accessible via a public snapshot GNSS online service. SnapperGPS allows everyone to build energy-efficient GNSS receivers at unmatched low costs, which is ideal for

applications like wildlife, fitness, or certain logistics tracking. The system was tested on data collected in a broad range of scenarios.

5

Deployments of Low-Cost Snapshot GNSS Receivers for Wildlife Tracking

5.1 Problem Statement

The introduction (Chapter 1) postulates that a low-power, low-cost, open-source snapshot GNSS receiver for wildlife tracking would benefit:

1. Longer deployments on small animals that cannot carry bulky batteries,
2. Studies with larger sample size on the same budget,
3. Uptake of wildlife tracking by conservationists and researchers for whom costs were previously too high, especially, in the Global South,
4. Tracking of aquatic animals that surface only briefly, and
5. Customisation of tags to cater specific deployment needs.

This chapter strives to (i) evaluate all five hypotheses using the example of SnapperGPS and (ii) validate that SnapperGPS as described in Chapter 3 and Chapter 4 is such a low-power, low-cost, open-source snapshot GNSS system that can be used by practitioners in the field.

This is done in two ways: firstly, analyses of selected deployments of SnapperGPS receivers by or in collaboration with third parties (Section 5.2), and, secondly, a quantitative study via a standardised survey involving past, current, and potential future SnapperGPS users (Section 5.3).

5.2 Selected Deployments

The examples in this section are a selection of SnapperGPS deployments. Descriptions highlight their relationships with hypotheses 1–5.

5.2.1 Tracking of a Large Population of Sea Turtles

In summer 2021 and summer 2023, loggerhead sea turtles were tracked in the Atlantic Ocean next to Alcatraz on Maio, Cabo Verde. The project was a collaboration with the Maio Biodiversity Foundation (FMB), a Cabo Verdean conservation organisation, and the Arribada Initiative, a UK community interest company specialising in conservation technology. Loggerhead sea turtles are classified as *vulnerable* on the Red List of the International Union for Conservation of Nature and Natural Resources (IUCN) and the Cabo Verdean sub-population is categorised as *endangered* [119, 120]. Maio, as a UNESCO biosphere reserve, is an important nesting site for them, with almost 20,000 nests per season on average [120]. Normally, the turtles only come ashore to nest and deposit eggs. They usually lay four egg clutches per season, separated by 12–17 days and at the same or near the same beach [120]. This renders recovery of sea turtle tracking tags possible when deployed during the nesting season. The FMB was interested in monitoring sea turtle movement during the nesting season to assess the effectiveness of the layouts of marine protected areas as well as to compare with the routes of fishing boats and industrial vessels since both are a primary threat to sea turtles. Since a larger number of tags was necessary to draw statistically significant conclusions, hypothesis 2 was relevant for the FMB, a local conservation organisation with a limited budget. Hypothesis 4 is also relevant for this deployment, since briefly surfacing aquatic animals were tracked.

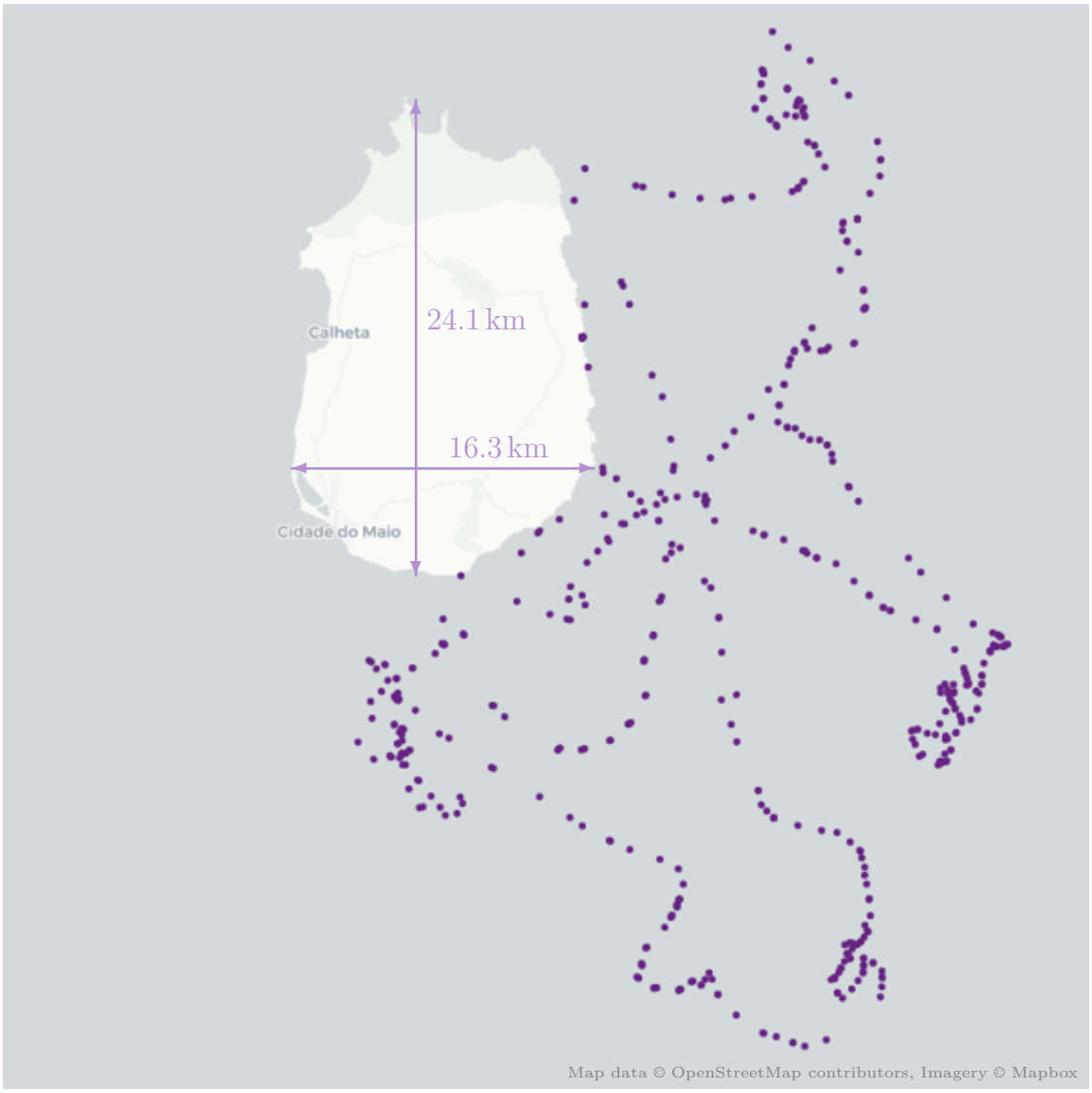


Figure 5.1: Exemplary track of a loggerhead turtle captured using the SnapperGPS system in the Atlantic Ocean during a period of two weeks in 2021.

In 2021, twelve SnapperGPS tags were deployed. The PCB version was similar to V1.0.0 (Chapter 4), but operated on LR44 coin cells. The SnapperGPS PCBs were connected to an external module to trigger snapshot capture based on resistance measurements or combined resistance and capacitance measurements between two electrodes on the outside of the tag[1] (Chapter 4).

However, the trigger module did not work most of the time, likely due to waterfilms and salt on the enclosure's surface. Therefore, snapshots could usually

[1]The resistance-based surfacing trigger module was not developed as part of this thesis.

Figure 5.2: Tracks of nine loggerhead turtles captured using the SnapperGPS system in the Atlantic Ocean during a period of two weeks in 2023.

only be captured in fixed intervals rather than when surfacing was detected. This still allowed to collect snapshots, but with a lower time resolution, as shown in Figure 5.1. The results verify that the SnapperGPS system is able to achieve location fixes from data collected during the brief surfacing events of sea turtles and to operate in a harsh environment, including sufficient battery life.

In 2023, 20 tags were deployed on and ten recovered from loggerhead sea turtles on Maio. This time, SnapperGPS tags with contact-less capacitance-based surfacing detection as seen in Figure 4.14 were employed. For packaging, enclosures made of a polyoxymethylene (POM) top cover and a POM or aluminium bottom plate were designed and used together with nitrile o-rings, cf. Figure 4.14. Prior to deployment, they were experimentally pressure tested to be waterproof to at least 120 m water depth. In addition, nine SnapperGPS receivers were deployed on and recovered from fishing boats, collecting data for a little more than six weeks, respectively.

In 2023, the trigger module was able to detect slightly more surfacing events than the one deployed in 2021, but the vast majority of snapshots was still captured in fixed intervals. It seems that even the contact-less surfacing detection method

is unable to perform reliably during extended durations in a harsh saltwater environment such as the Atlantic. A marine paint and/or coating could potentially have improved results. Despite the issues with the surfacing detection, the temporal resolution of eight of the nine tracks is high enough to give a clear picture of the movement of the turtles, see Figure 5.2. The tenth tag did not record data, potentially due to a user error during configuration.

In Detail: How to Deploy a Sea Turtle Tag
Consumables:

- Woven fibreglass mats. For a single tag, five pieces need to be cut. One that is slightly larger than the footprint of the tag and four that are slightly longer than the sides of the tag.
- Fast hardening metal glue/putty/adhesive (compound that contains metal), e.g., LOCTITE Metal Magic. Ideally, it should come as a kneadable stick and harden/set/cure within 5–10 min. About 15–30 g are needed per turtle.
- Epoxy paste/adhesive in cartridges, e.g., from DEVCON. It should harden in a few minutes.

Equipment:

- Pair of scissors (to cut fibreglass mats)
- Red head lamps (for visibility at night without disturbing turtles)
- Cloth, maybe sanding paper (to clean part of carapace)
- Epoxy gun (if the epoxy comes in cartridges)
- Spatulas (to prepare and apply epoxy)
- Cups/bowls (to prepare epoxy)
- Optionally gloves (when working with the metal glue and epoxy)
- Knife with a long blade (to remove fibreglass and tag after deployment)

Steps before deployment (day):

- Configure SNAPPERGPS receiver.
- Cut fibreglass mats into rectangles that will hold the tag on the carapace; also cut smaller pieces for the sides of the tag.

Steps during deployment (night):

- Use red head lamps to not disturb turtles.
- Find a nesting turtle.
- Wait until she has started laying eggs.
- Clean top part of carapace, specifically, second scute from the front.
- Lay down fibreglass mat there.
- Use epoxy gun to squeeze epoxy into cup/bowl.
- Prepare epoxy by mixing with spatula.

- Apply epoxy to fibreglass square to fix it to the carapace.
- Wait until epoxy is hard (∼5 min).
- Prepare/knead metal glue.
- Apply metal glue to bottom of enclosure.
- Glue enclosure on fibreglass.
- Apply metal glue to sides of enclosure to seal gaps between carapace and enclosure.
- Apply smaller fibreglass pieces to sides of enclosure; fix with more epoxy.
- Leave turtle.

Steps two weeks later (night):

- Patrol beach when turtle is expected to return.
- When tagged turtle is found, wait until she has started laying eggs.
- Remove fibreglass and tag with knife.
- Read out data; upload to SnapperGPS server.

5.2.2 Tracking of Sea Turtles on a Small Budget

Figure 5.3: Left: A Núi Chúa National Park employee with SnapperGPS turtle tags. (Image courtesy of Meredith S. Palmer.) **Right:** A green turtle with a SnapperGPS tag in the Núi Chúa National Park. (Image courtesy of Tran Viet Phong.)

In 2023, green sea turtles were tracked in the South China Sea next to the Núi Chúa National Park (NCNP), Vietnam. This project was a collaboration with Fauna & Flora International (FFI), an international conservation organisation, and the NCNP, cf. Figure 5.3. Green turtles are also listed as *endangered* on the Red List of the IUCN [119] and behave similarly to loggerhead sea turtles (Section 5.2.1). The population in Núi Chúa is small with less than ten individuals nesting every year. This is the last known sea turtle population that nests on

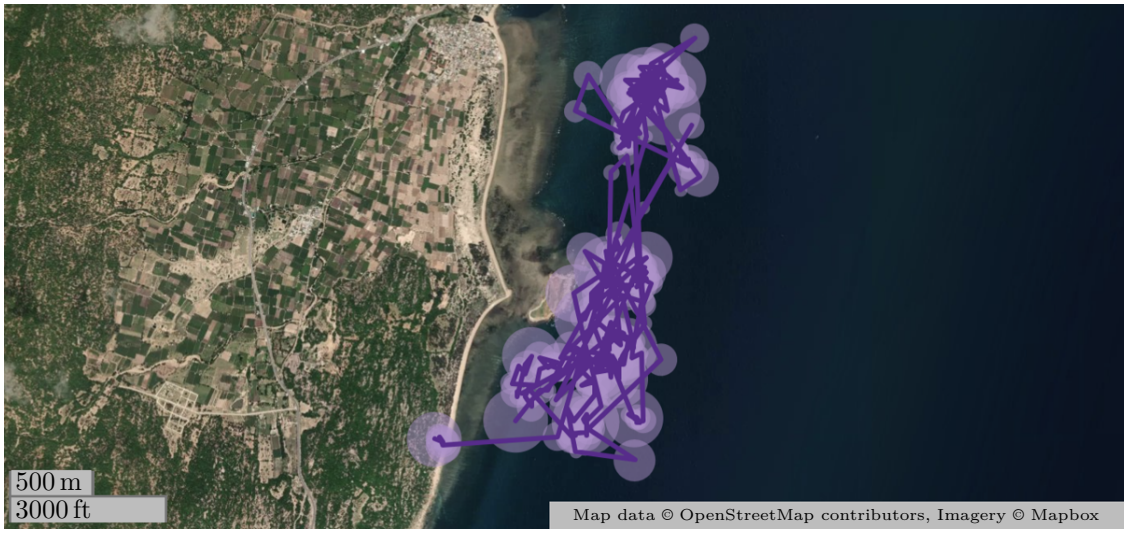


Figure 5.4: Exemplary track of a green turtle captured using the SnapperGPS system in the South China Sea during a period of two weeks. Pink: location fixes with uncertainties as circles. Purple: polyline connecting the fixes. Printed with permission of the data owners.

the beaches of mainland Vietnam, rendering it a primary target of conservation action. The project partners were looking for turtle tags to monitor movement and compare it with the boundaries of the local marine protected area. They had not worked with tracking technology before due to its high cost. In consequence, hypotheses 3 and 4 are relevant for this deployment.

Out of five deployed tags, three were recovered after about two weeks. Two did not collect data, likely due to user mistakes (one deployed without being configured and one deployed with drained batteries). For the third one, surfacing detection did not work for unclear reasons. In consequence, the tag was again only able to collect data with a low time resolution in fixed intervals, see Figure 5.4. However, the resolution was high enough for the target use case.

Deployments are scheduled to continue in 2024.

5.2.3 Long-Term Tracking of Terrestrial and Freshwater Turtles

Several SnapperGPS tags were deployed by the University of Florida to track multiple terrestrial and freshwater turtle species in the USA, including Gopher tortoises and Suwannee alligator snapping turtles. Deployment durations ranged

Figure 5.5: A Gopher tortoise with a SnapperGPS receiver. (Images courtesy of Travis M. Thomas.)

from several weeks to several months, requiring low-power tags (hypotheses 1). Standard SnapperGPS receivers (V1.0.0, Chapter 4) were deployed on Gopher tortoises, who are terrestrial tortoises that spend their time on land or in burrows, see Figure 5.5. Individuals of these endangered species were relocated from an air force base to an surrogate habitat. A goal of the SnapperGPS deployment was to assess whether the individual would persistently stay at or move away from its release location and to estimate its home range. SnapperGPS was primarily chosen because of its affordability (hypothesis 2) and long battery life (hypothesis 1), with being open-source being an additional advantage (hypothesis 5). High localisation accuracy was not necessary for this deployment. The two primary competing technologies for this type of deployment are (i) conventional GPS tags, which are more expensive and require larger batteries but provide higher localisation accuracy, and (ii) VHF tags, which come with high labour costs in high-income countries such as the USA but battery lives of at least six to eight weeks.

On the Gopher tortoises, the receivers were deployed in heat-shrink wrapping. Other deployments used resin housings created by the end user and also involved accelerometer daughterboards, two examples of customisation (hypothesis 5).

All recovered tags collected data over their whole deployment duration. No quality decrease of the location estimates over time was visible, despite repeated intermediate losses of sky view inside of burrows and temperature differences between night and day of about 20 °C.

For some of the freshwater deployments, hypothesis 2 was relevant, since enough individuals should be tracked to draw statistically significant conclusions, while recovery rates were estimated to be only between 50% and 100%, rendering tracking tags that cost several hundred or even several thousand dollars uneconomically.

Deployments of some tags are still on-going as of writing.

5.2.4 Non-Obtrusive Tracking of Birds

The largest amount of GNSS snapshots that were collected during animal tracking studies and processed by the SnapperGPS software comes from sea birds. For example, third-party snapshot GNSS receivers compatible with the SnapperGPS app were deployed by the University of Oxford on Manx shearwaters while foraging in the Irish Sea in spring 2022 [121]. The reason for choosing snapshot GNSS was to achieve a high sample size on a limited budget to make inferences about species behaviour [121] (hypothesis 2). In addition, the tags needed to be light enough to be carried by the birds (weighting about 500 g) over long distances of hundreds of kilometres (hypothesis 1).

At least 19 tracks were successfully recorded this way in 2022 [121] and many more in 2023.

5.2.5 Cost-Efficient Tracking of Goats

In 2022, domestic goats were tracked on Brava, Cabo Verde. The project partner, FFI, was looking for a cost-effective way to enable local herders to track their goats. Conservationists would then overlay the tracks with vegetation maps. Hypotheses 2 and 3 were primarily relevant for this deployment.

Several tracks were recovered, including those in Figure 5.6.

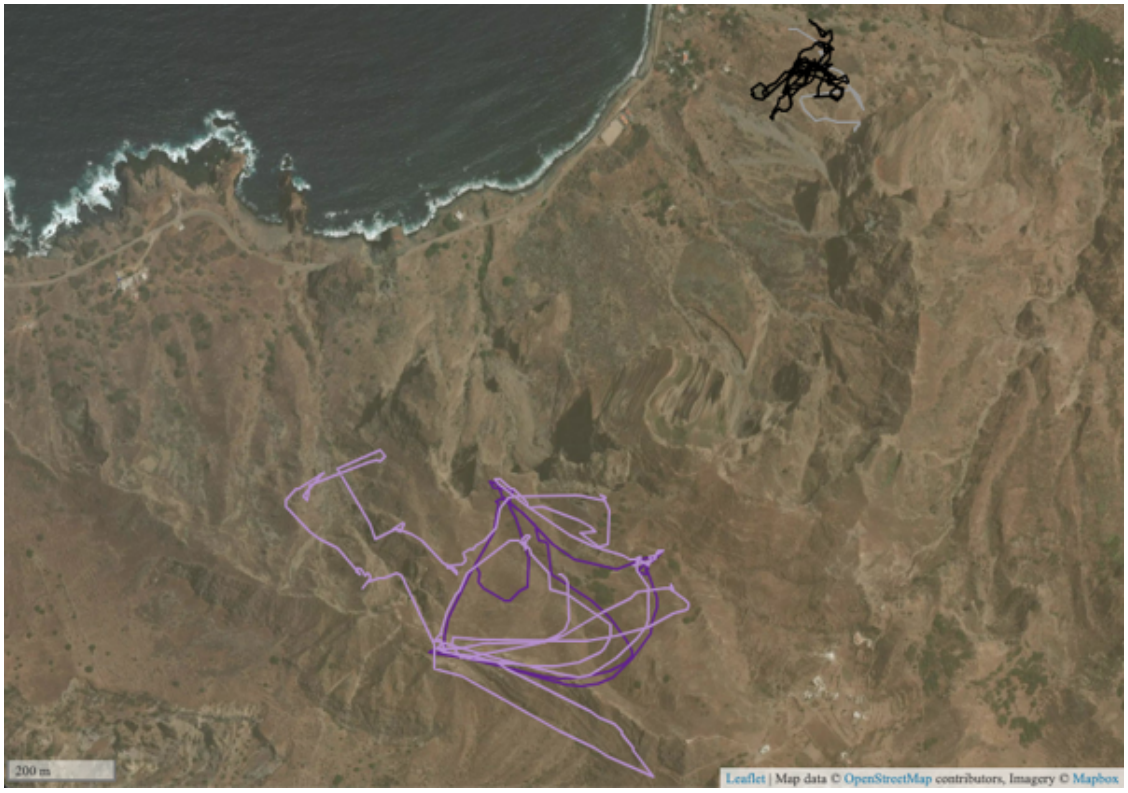


Figure 5.6: Four tracks of goats captured by Fauna & Flora International and the Associação Biflores using the SnapperGPS system on Brava, Cabo Verde. Printed with permission of the data owners/creators.

5.2.6 Research and Development Platform

In light of hypothesis 5, at least six SnapperGPS users built their own tags, e.g., for commercial purposes or for research. This includes a university where SnapperGPS receivers were used in a project aiming to develop a battery-free snapshot GNSS receiver.

5.3 User Feedback

Between summer 2021 and summer 2023, more than 150 SnapperGPS receivers were distributed to around 30 different parties[2], mainly conservationists and academic researchers. In addition, at least five individuals independently replicated

[2]Not including around 100 boards of a different version designed by Amanda Matthes, Andy Hill, and Alex Rogers that were also used with with the firmware, web app, and algorithms presented in this thesis.

and used the open-source SnapperGPS hardware with success[3]. Furthermore, about 20 parties expressed written interest in SnapperGPS, but neither were selected to receive hardware, nor are known to have replicated the hardware themselves as of writing.

In 2023, all contact persons were sent a standardised anonymous survey form[4] asking them questions related to why they were interested in SnapperGPS, what they think about SnapperGPS, details about their deployments, and user demographics to further evaluate the five hypotheses 1–5 set out in Chapter 1 and Section 5.1. In total, 28 people responded.

The following Section 5.3.1 aims to establish the overall reasons why individuals were interested in SnapperGPS. It is followed by five sections that individually evaluate each of the five hypotheses 1–5. Finally, Section 5.3.7 covers suggestions for improvements and future work provided by users and addresses the question whether SnapperGPS is perceived to fulfil its design goals or not.

5.3.1 Overall Interest

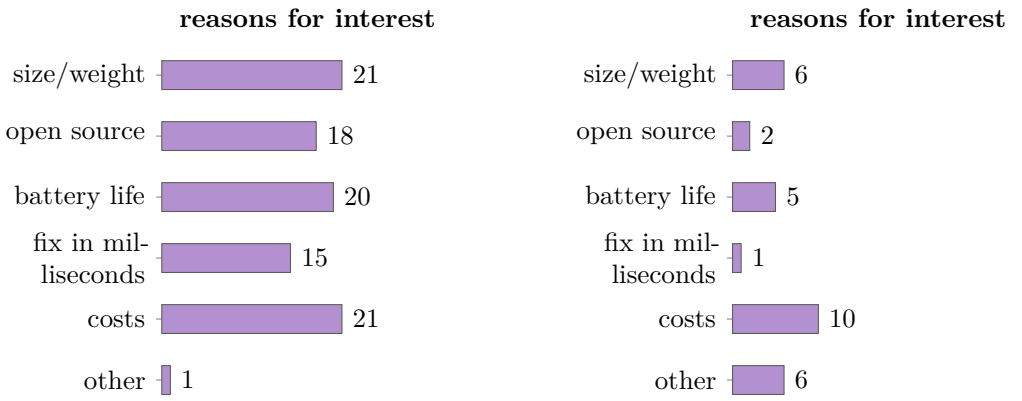
A central question in the structured survey spanning all five hypotheses is:

? Question 3: "Why are you interested in SnapperGPS?"

Figure 5.7a presents the answers. It is important to note that all respondents belong to the group of individuals who were/are interested in SnapperGPS. People who had no or little interest are not represented. The results show that most participant's interest originated from a mix of factors rather than an individual one. Each individual point was relevant to more than 50% of the respondents with "SnapperGPS's ability to acquire a fix within milliseconds, e.g., during brief surfacing of a marine animal." being the least relevant one (54%) and "SnapperGPS's size/weight" and "SnapperGPS's costs" being the most relevant

[3]Three independent reports of individuals who replicated SnapperGPS themselves can be found on `https://github.com/SnapperGPS/snappergps-pcb/discussions/18`, `https://github.com/SnapperGPS/snappergps-pcb/discussions/26`, and `https://github.com/orgs/SnapperGPS/discussions/14`, respectively.

[4]Ethics approval reference number: C1B-23HT-COML-002.



(a) Responses to question 3: "Why are you interested in SnapperGPS?" ($n = 28$).

(b) Stated reasons for interest in SnapperGPS extracted from submitted project proposal texts ($n = 50$).

Figure 5.7: Questions and responses related to the overall interest. "Other" includes "reusing the positioning software" (1), accuracy" (1), "snapshot GNSS" (2), "high-frequency tracking" (2), and "new technology" (1).

ones (75%). Although, the latter might be biased since some users received the hardware for free, co-financed, or on loan. In consequence, some of these users might have responded based on their actual costs rather than taking the hypothetical $50 per device into account that are mentioned in the survey.

The question was presented with pre-defined multiple-choice answers. In such a setting, participants are likely to tick answer options with which they would not have come up themselves if they would not have been presented with them. Therefore, the results are cross-validated by analysing the texts of 50 received SnapperGPS project proposals regarding the stated reason of interest. Figure 5.7b shows the results. In general, people stated less reasons per submission in their textual descriptions than the participants did in the questionnaire. However, the overall picture is similar, with costs, battery life, and weight/size being the most important factors. The relative count for "fix in milliseconds" is much lower. However, it is likely that most people who were interested in tracking of aquatic animals were interested in this feature, but just did not state it explicitly in their proposal, see Section 5.3.5.

In general, the answers from Figure 5.7a and Figure 5.7b map to the hypotheses 1–5, see Table 5.1. This suggests that all hypotheses are perceived as valid by an existing and potential user base.

#	hypothesis	answers in Figure 5.7
1	Longer deployments on small animals	size/weight; battery life
2	Studies with larger sample size on the same budget	costs
3	Uptake of wildlife tracking by conservationists and researchers for whom costs were previously too high	costs
4	Tracking of aquatic animals that surface only briefly	fix in milliseconds
5	Customisation of tags to cater specific deployment needs	open source

Table 5.1: Mapping of hypotheses 1–5 to answers in Figure 5.7.

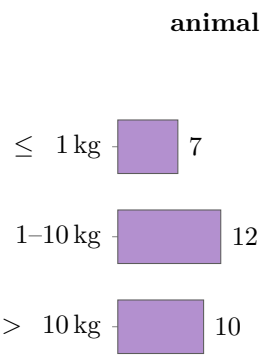
In the following, questions and answers related to each hypothesis individually are evaluated.

5.3.2 Longer Deployments on Small Animals

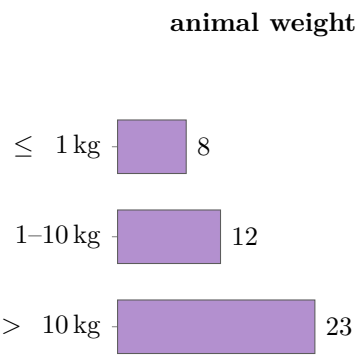
The questionnaire contains three questions that indirectly provide insights whether the participants considered the technology as suitable for longer deployments on small animals or not:

- *?* Question 11: "What are your species of interest?"
- *?* Question 17: "Regarding its size and weight, SnapperGPS is ... than other trackers that could be used for your target deployment."
- *?* Question 18: "SnapperGPS's battery life is ... than other trackers that could be used for your target deployment."

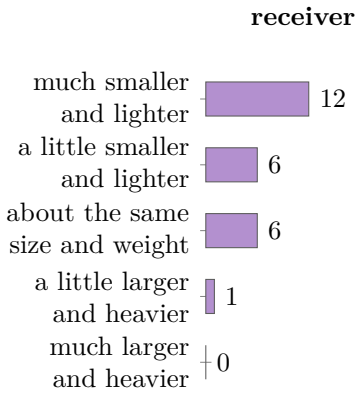
For question 17 and 18, participants were asked to complete the sentence with one of five options, cf. Figure 5.8c and Figure 5.8d. In addition, 50 written project proposal texts were analysed to establish the species of interest. Together with the



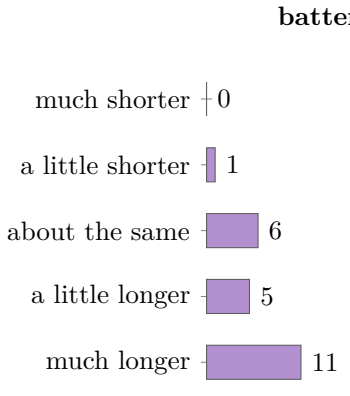
(a) Histogram of average weights of animals mentioned in answers to question 11: "What are your species of interest?" ($n = 26$).



(b) Histogram of average weights of animals of interest extracted from submitted project proposal texts ($n = 44$).



(c) Responses to question 17: "Regarding its size and weight, SnapperGPS is . . . than other trackers that could be used for your target deployment." ($n = 25$).



(d) Responses to question 18: "SnapperGPS's battery life is . . . than other trackers that could be used for your target deployment." ($n = 23$).

Figure 5.8: Questions and responses related to hypothesis 1: *longer deployments on small animals.*

answers to question 11, they were mapped to the average weight of an individual of this species [122, 123], cf. Figure 5.8a and Figure 5.8b.

The answers to question 17 are evidence that the majority of participants (72%) perceive SnapperGPS as smaller and lighter than competing hardware and—since hardware size and weight are the most important factors for determining whether a tag can be deployed on a small species—directly support hypothesis 1. This weight/size advantage is not perceived as jeopardising deployment durations, with 96% of the participants responding to question 18 that SnapperGPS's battery life is not shorter than the one of competing technologies and even 70%

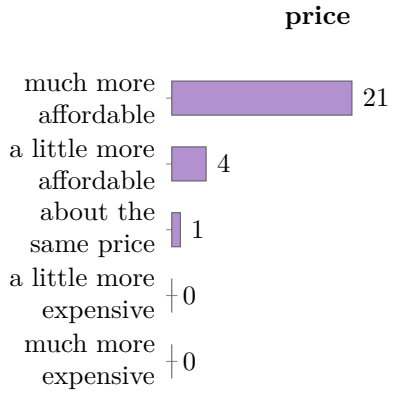
saying that it is longer.

In addition, Figure 5.8a shows that 27% of the conservationists and researchers that participated in the survey planned to deploy SnapperGPS on smaller animals weighting less than 1 kg. Based on the rule of thumb that a tracker should not weight more than 5% of an animal's body mass (see Section 2.14), this requires trackers weighting 20 g or less, which excludes all commercial trackers in Table 2.3 and DIY trackers in Table 2.4 (that can operate for more than a day), but does not exclude SnapperGPS hardware, which weights around 10 g, see Chapter 4. Most participants would like to deploy on animals in the 1–10 kg range, which allows the use of some, but not all trackers in Table 2.3 and Table 2.4, while only a minority considers animals with larger body mass, which basically allows to attach any listed tag. While the bias towards smaller and medium rather than larger species in the survey suggests the relevance of SnapperGPS's property of being comparably light and small, this is not as visible in the project proposal texts (Figure 5.8b) with less than half the species weighting less than 10 kg. However, this is partially due to the large number of submissions targeting tracking of surfacing aquatic species, which exclusively fall into the heaviest category.

The SnapperGPS receivers are not among the very smallest and lightest existing GNSS receivers for wildlife tracking, which can weight as little as 1.3 g [15]. This is, firstly, because of its discrete design requiring more PCB area than a receiver with an integrated GNSS module (IC) and, secondly, the larger and heavier patch antenna to provide sufficient signal quality in the short GNSS snapshots. So, for very short deployments, other GNSS receivers offer a weight advantage. However, the aforementioned 1.3 g tag, for example, runs less than a day on its 30 mA h battery while recording around 10,000 fixes. With the same battery, SnapperGPS could collect twice as many datapoints over the course of a year.

5.3.3 Studies with Larger Sample Size on the Same Budget

An important factor for the number of deployable tags is the cost associated with sourcing an individual tag. Hypothesis 3 is concerned with this and the



(a) Responses to question 22: "At a price of $50 (including battery and antenna, excluding enclosure and tax/duties), SnapperGPS would be ... than other trackers that could be used for your target deployment." ($n = 26$).

Figure 5.9: Question and responses related to hypothesis 2: *studies with larger sample size on the same budget.*

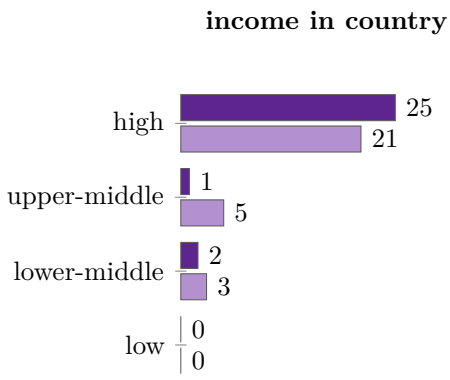
questionnaire's most relevant question is:

? Question 22: "At a price of $50 (including battery and antenna, excluding enclosure and tax/duties), SnapperGPS would be ... than other trackers that could be used for your target deployment."

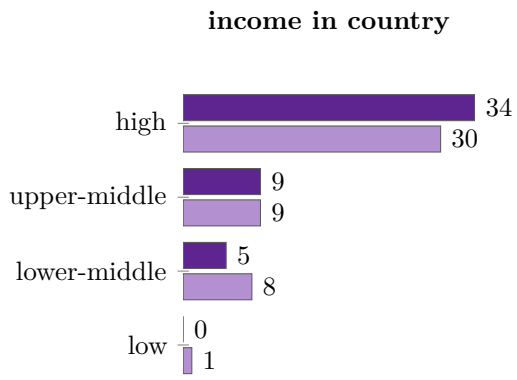
The component cost of a single SnapperGPS receiver (in a batch of 100) is less than $30, cf. Section 4.2.8, but the higher price of $50 was chosen for the questionnaire to account for distribution-related costs. Still, 96% of the respondents perceive SnapperGPS as "more affordable", with 81% saying that it is "much more affordable" (Figure 5.9a).

5.3.4 Uptake of Wildlife Tracking by Conservationists and Researchers for Whom Costs were Previously Too High

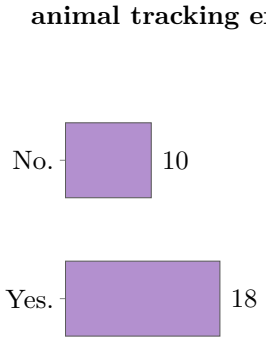
Section 5.3.3 provides evidence that SnapperGPS is perceived as a low-cost solution already. However, it does not reveal whether SnapperGPS (a) mainly offers cost savings for conservationists and researchers operating on a large budget or (b) makes tracking accessible to users with a small budget, who may not be able



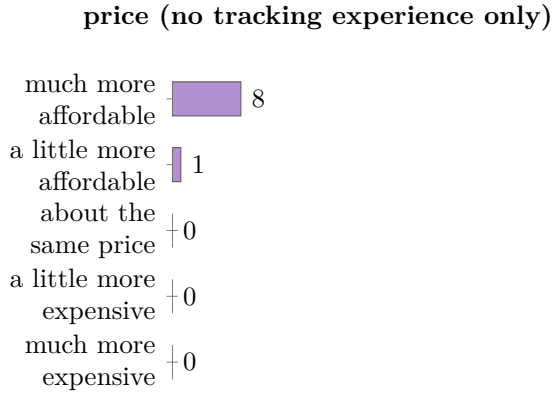
(a) Responses to question 13: "What are your target countries for deployments?" (pink) and question 35: "In which country are you employed?" (purple) grouped into low, lower-middle, upper-middle, and high income economies in 2023 according to the World Bank [124] ($n = 28$ and $n = 28$, respectively). Three answers ("Africa", "North America", and "worldwide") are excluded from the former bars.



(b) Countries of (planned) deployments (pink) and countries where submitters were based (purple) extracted from written project proposals and grouped into low, lower-middle, upper-middle, and high income economies in 2023 according to the World Bank [124] ($n = 50$).



(c) Responses to question 27: "Have you previously used GPS for animal tracking?" ($n = 28$).



(d) Responses to question 22: "At a price of $50 (including battery and antenna, excluding enclosure and tax/duties), SnapperGPS would be ... than other trackers that could be used for your target deployment." of participants who responded "No." to question 27: "Have you previously used GPS for animal tracking?" ($n = 9$).

Figure 5.10: Questions and responses related to hypothesis 3: *uptake of wildlife tracking by conservationists and researchers for whom costs were previously too high.*

to conduct tracking studies without access to a low-cost system. For this, a look at the user demographics is relevant. Users and potential users were asked about their deployment country/countries as well as the country they are based in:

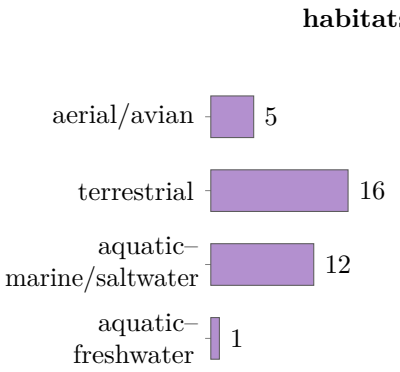
- Question 13: "What are your target countries for deployments?"
- Question 35: "In which country are you employed?"

The latter is known to be correlated with the financial resources available, with the Global South usually being disadvantaged in contrast to the Global North [124]. Figure 5.10a and 5.10b show a correlation with the GDP per capita of the respective country, revealing that the majority of (prospective) users is still based in high income economies. An issue hindering further adoption of SnapperGPS in low and middle income economies could have been that information on SnapperGPS was only available in English. However, if wildlife conservation work or research is led by institutions and/or individuals from high income countries, local conservationists or researchers may still be involved, but not visible in the presented statistics. Some nations even mandate collaborations with local scientists [125, 126].

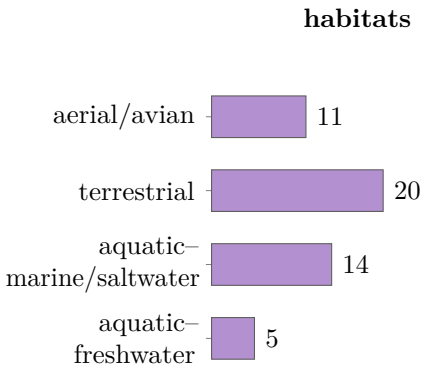
In addition, three survey questions relate to the participants' previous use of similar technologies:

- Question 27: "Have you previously used GPS for animal tracking?"
- Question 28: "Have you previously ever done any animal movement data analysis?"
- Question 29: "Have you ever deployed sensors on animals before?"

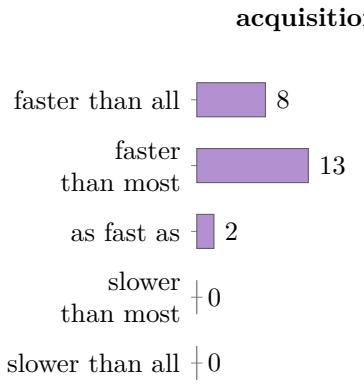
The answers (Figure 5.10c) reveal that SnapperGPS was able to attract a number of parties (36% of the respondents) that did not use GPS for animal tracking previously. Noteworthy, Figure 5.10d reveals that almost all of these respondents (89%) also said that SnapperGPS is "much more affordable" than competing systems, suggesting that they may not have worked with tracking technology before because of its costs.



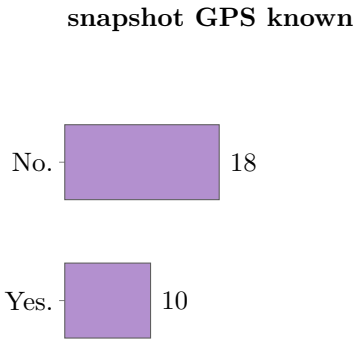
(a) Responses to question 12: "What are your habitats of interest?" ($n = 26$).



(b) Deployment habitats extracted from written project proposals ($n = 50$). For five submissions, the tracking habitat was either not clearly defined or a use case not related to animal tracking was proposed. Four submissions proposed multiple deployments in different habitats.



(c) Responses to question 19: "SnapperGPS's ability to acquire the data for a fix within twelve milliseconds is ... other trackers that could be used for your target deployment." ($n = 23$).



(d) Responses to question 30: "Did you hear about the "snapshot" GPS technique before you learned about SnapperGPS?" ($n = 28$).

Figure 5.11: Questions and responses related to hypothesis 4: *tracking of aquatic animals that surface only briefly.*

5.3.5 Tracking of Aquatic Animals that Surface Only Briefly

A unique property of snapshot GNSS is its ability to acquire the satellite data necessary for fix within milliseconds, enabling the tracking of aquatic animals with surfacing times of less than a few seconds, which is not achievable with traditional GNSS techniques. Three questions in the questionnaire aim to understand whether this feature was relevant to (potential) users or not:

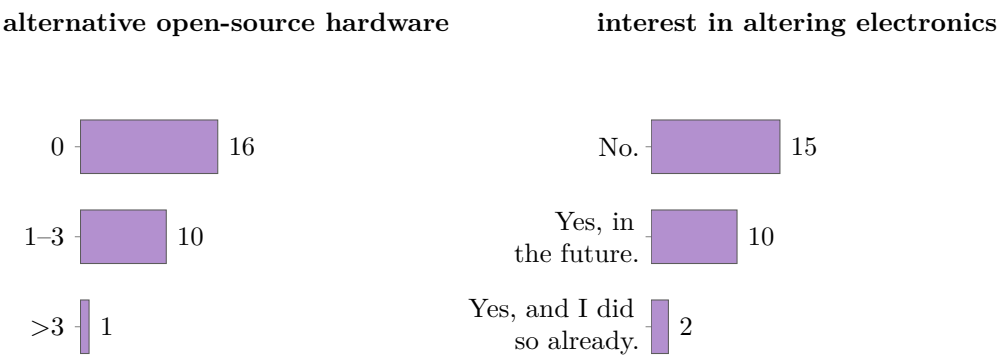
- ? Question 12: "What are your habitats of interest?"
- ? Question 19: "SnapperGPS's ability to acquire the data for a fix within twelve milliseconds is ... other trackers that could be used for your target deployment."
- ? Question 30: "Did you hear about the "snapshot" GPS technique before you learned about SnapperGPS?"

In addition, the tracking habitat was also extracted from 50 project proposal texts and the results for all four are shown in Figure 5.11a, 5.11b, 5.11c, and 5.11d.

First, the responses to question 19 clearly demonstrate that almost all participants (91%) were aware that most wildlife tracking tags do not acquire the data for a fix within milliseconds, unlike when snapshot GNSS is employed. Figure 5.11a and Figure 5.11b show that tracking of aquatic animals represents almost 40% of the proposed deployments. This is despite the fact that only the bare-bone open-source SnapperGPS boards presented in Figure 4.1 were available to almost everyone rather than a dedicated aquatic tag (like one of the tags shown in Figure 4.14), burdening potential users with coming up with the housing and triggering solutions themselves and hence almost certainly reducing interest in aquatic tracking. Finally, Figure 5.11d reveals that the snapshot GNSS technique was already known by 36% of the participants before they learned about SnapperGPS, despite this technique being rarely used in any wildlife tracking tags except for a few models designed for marine and avian tracking, cf. Section 2.7 and 2.15. This suggests that a significant number of parties were interested in SnapperGPS because of this feature in particular.

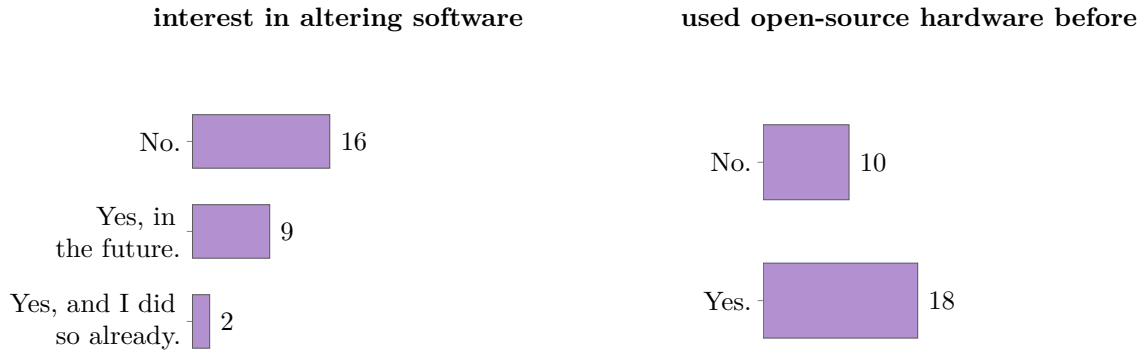
5.3.6 Customisation of Tags to Cater Specific Deployment Needs

A number of survey questions is related to the fact that SnapperGPS is open source and its implications, including the opportunity to customise tags for deployments,



(a) Responses to question 21: "Besides SnapperGPS, of how many open-source hardware projects are you aware of that could be used for your target deployment?" ($n = 27$).

(b) Responses to question 23: "Are you interested in altering the SnapperGPS electronics hardware yourself?" ($n = 27$).



(c) Responses to question 24: "Are you interested in altering the SnapperGPS firmware or processing software yourself?" ($n = 27$).

(d) Responses to question 31: "Have you ever used open-source hardware before?" ($n = 28$).

Figure 5.12: Questions and responses related to hypothesis 5: *customisation of tags to cater specific deployment needs.*

which is usually not possible for users of commercial tags, at least not without (potentially costly) support of the manufacturer:

- *?* Question 21: "Besides SnapperGPS, of how many open-source hardware projects are you aware of that could be used for your target deployment?"
- *?* Question 23: "Are you interested in altering the SnapperGPS electronics hardware yourself?"
- *?* Question 24: "Are you interested in altering the SnapperGPS firmware or processing software yourself?"

- ? Question 31: "Have you ever used open-source hardware before?"

Figure 5.12a reveals that most participants (59%) were not aware of any alternative open-source solution for their deployments and almost nobody was aware of a large number of more than three (4%). In contrast, Figure 5.12b and Figure 5.12c show that about half of the participants were interested in customisation of tags (44%) and/or the software (41%). This highlights that there is a strong interest in open-source wildlife tracking tags such as SnapperGPS and that SnapperGPS is perceived to be suitable for end-user customisation, both, its software and its hardware.

Figure 5.12d shows that SnapperGPS appeals to both, people without and with prior experience with open-source hardware (36% and 64% of the participants, respectively). The interest of the first group (people responding "No.") highlights SnapperGPS' potential as a door opener for the adaption of open-source conservation technology in general.

5.3.7 Problems and Suggestions for Future Work

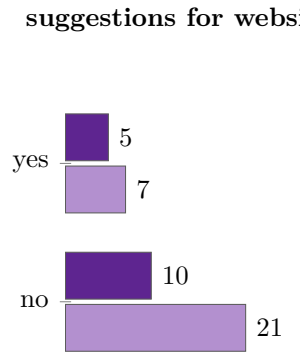
The previous sections analyse whether a system as SnapperGPS is described to the public fulfils hypotheses 1–5. However, it is worthwhile to ask whether the SnapperGPS implementation actually fulfils the users' expectations. Therefore, question 14, question 15, and question 16 aimed to reveal any issues users might have had by asking them for improvement suggestions:

- ? Question 14: "Do you have any suggestions for the SnapperGPS app or website?"
- ? Question 15: "Do you have any suggestions for the SnapperGPS hardware?"
- ? Question 16: "Do you have any suggestions for the SnapperGPS documentation?"

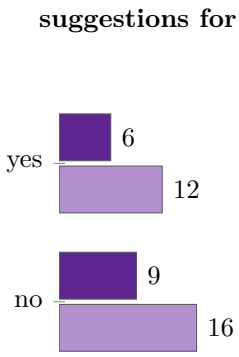
These questions were optional and 16, 21, and 15 individuals responded, respectively. However, a substantial amount of responses were not actually suggestions,

	suggestion
app/website	• The website looks great for how to use the product. But more information on the product itself, prior to receiving them/deciding to get them, would be great! • Maybe a How-to for beginners • Overall, the website is great! Adding instructional videos on assembly, troubleshooting, and best practices might be helpful. Maybe have the option for users to create a profile that can list all previous upload IDs and maybe map multiple tracks in the browser. • Great user friendly website! • It did precisely what I needed for the few times I had to use it. The interface is easy to navigate and the processing is rather fast. I don't remember if it's already the case: but you could add arrows to see the direction of movement between nodes. • Website is nice, intuitive and easy to use! • I like the homepage layout, and how it describes the workflow as well as links to common issues. I think that is really helpful to people. The website itself was straightforward and easy to use. I am not sure if this is possible, but having a page where you can compare two tracks or upload data from two separate deployments would be helpful • Everything worked fine. I had an issue with my antenna, but the website was easy to use. • The site is easy to use [...], but did not work well inside building. • We found the website straightforward to navigate and easy to train staff on. There are plenty of helpful resources and instructions are presented in clear, accessible language. • On the Workflow section of the homepage, where it says "Get/build your SnapperGPS receiver", it could be helpful to directly include a link to where to buy receivers or their components (e.g. the `https://github.com/SnapperGPS/snappergps-pcb#getting-started` page)
hardware	• We can add new radio transceiver • Smaller is always better • For Manx shearwaters, a narrower board was better than the standard board. We also faced some issues with the unevenness of the components on one side, creating a lopsided deployment that we corrected with layers of thin plastic board to make balance the asymmetric components. • Different configuration options would be wonderful. Having multiple options to choose from (e.g., square, round, long rectangle) gives Biologists more options to attach the SnapperGPS units to different species and size classes. • The implementation of a bluetooth download versus hardline download would be great, particular to limit the need to physically handle the animal for longer than needed (problematic with venomous snakes) • Sell the pre assembled device :) • Integration with an immersion logger would be useful for seabirds (detection of saltwater immersion). In the long term it would be very useful to have a device that could record locations at a high resolution (<=10 sec) for several months, potentially coupled with an immersion logger (eg usually 1 min resolution, but when the logger gets wet, change to 10 sec resolution for 1 hour). • The lighter you can make it, the more interested we are! • Hardware is small, lightweight and works properly. • I think the instructions given with each device made using the hardware easy for someone who does not know a lot about GPS devices or this type of hardware in general. • Further miniaturisation. Increase sample frequency from 4 MHz to 16 MHz or higher. Integrate a chip antenna. Create a "motherboard" that offloads some of the space and battery-using components (USB-C connector and LEDs come immediately to mind). The motherboard could also contain a full GNSS receiver and provide the tag with an initial position during deployment. This would eliminate the step of selecting an initial position during data upload. An "is submerged" sensor. Save power by not attempting to capture data when underwater. Switch from Eagle to KiCad (open-source EDA). • Make it smaller? easy to ask but I'm afraid that the hardware is a bit too large for a feral cat • We ran into a few issues with batteries draining because staff were unaware whether device was on or off or how charged the battery was: any way to more intuitively indicate status of the battery would be helpful. • Width of the board to be narrower to better fit on a collar ∼15 mm wide
documentation	• Clarification of the "error" calculation on the website, I was provide this by Jonas but would be good for the website. • I really appreciate a documentation of this level of quality. • I mostly worked with snapshot-gnss-algorithms-main. There are a lot of scripts where it takes time to get an overview of the code structure and dependencies. It would have been helpful to have an a bit more extensive documentation there but the most important points are documented. I am thankful that everything is open-source! I want to implement parts of snappergps-backend. The documentation there is clear. Maybe you could add some statements about platform support (is windows supported?). • Again, this is easy to understand. I like how you can download a file that integrates the data into googleMaps. It was a little hard to understand what the difference between the pale coloured markers and the bright-coloured markers was (I assume confidence?). • It is spread out a little bit, but I think very well done overall. • SnapperGPS documentation is straightforward and accessible.

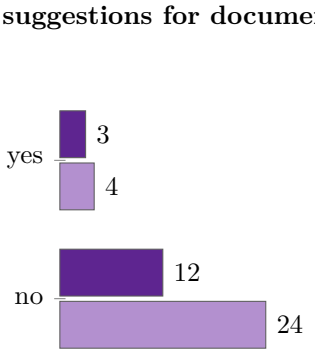
Table 5.2: Responses to question 14: "Do you have any suggestions for the SnapperGPS app or website?", question 15: "Do you have any suggestions for the SnapperGPS hardware?", and question 16: "Do you have any suggestions for the SnapperGPS documentation?" that are not "no" or "N/A" etc.



(a) Responses to question 14: "Do you have any suggestions for the SnapperGPS app or website?" (pink: all participants $n = 28$, purple: participants with physical access to SnapperGPS receivers $n = 15$).



(b) Responses to question 15: "Do you have any suggestions for the SnapperGPS hardware?" (pink: all participants $n = 28$, purple: participants with physical access to SnapperGPS receivers $n = 15$).



(c) Responses to question 16: "Do you have any suggestions for the SnapperGPS documentation?" (pink: all participants $n = 28$, purple: participants with physical access to SnapperGPS receivers $n = 15$).

Figure 5.13: Questions and responses related to suggestions for future work.

but positive feedback (despite participants not being asked for positive comments). Therefore, Figure 5.13a, Figure 5.13b, and Figure 5.13c show how many participants entered actual suggestions. In addition, Table 5.2 lists all responses.

Most of the participants had no suggestions at all and none of the participants suggested that the core functionality of the system should be improved or bugs or other problems should be addressed. Instead, comments focus on improving documentation and adding features to the web application or hardware. This absence of negative comments towards the core system can likely not be explained

by users being hesitant to give negative feedback since the survey was anonymous. It can also likely not be explained by users having little experience with the system since some of the comments reveal familiarity with details of the workflow and the system architecture. In consequence, it is likely that SnapperGPS' core technology performed satisfactory for the users that participated in the survey and fulfilled their expectations. This is further supported by the large proportion of positive feedback, which was not expected in response to question 14, question 15, and question 16.

Most suggestions focus on improving documentation or adding—usually minor—features to the web application. Only suggestions regarding the hardware are more substantial, including different board shapes, increased memory, reduced weight, use of a chip antenna, detection of (salt-)water immersion, wireless data download, and a higher sampling rate. In response to the feedback, the first six points are addressed in Chapter 4 and Chapter 6 (use of a chip antenna not successfully, though). The last point remains as potential future work. Increasing the sampling frequency would likely make the system more complex and increase its energy and memory consumption, like for some of the existing snapshot GNSS systems covered by Section 2.7. On the other hand, it could improve signal quality, and, in consequence, enable the use of a lighter chip/PCB antenna.

Some of the suggestions for the hardware reveal interest in customised versions for specific species (narrow bird board, wireless snake board, aquatic board with immersion sensor...), adding further evidence for hypothesis 5.

5.4 Conclusions

The discussion of the SnapperGPS survey in Section 5.3 clearly shows that all five hypotheses 1–5 were perceived to be valid by a majority of the participants, respectively, and that SnapperGPS is perceived to be a system that addresses all five hypotheses. In addition, the deployments proposed and conducted by the participants of the survey clearly support hypotheses 1, 2, 4, and 5. Only for hypothesis 3 (*Uptake of wildlife tracking by conservationists and researchers for*

whom costs were previously too high, especially, in the Global South), there is limited support from the deployments reported in the survey.

The use cases covered in greater depth in Section 5.2 provide further evidence for all five hypotheses. Each of them was relevant to two or more of the deployments. Section 5.2 also validates that SnapperGPS is a system that can be used in practice to address all hypotheses.

To summarise, this chapter provides evidence that SnapperGPS is a low-power, low-cost, open-source snapshot GNSS system that benefits (1) longer deployments on small animals that cannot carry bulky batteries, (2) studies with larger sample size on the same budget, (4) tracking of aquatic animals that surface only briefly, and (5) customisation of tags to cater specific deployment needs. It also seems to have the potential to benefit (3) uptake of wildlife tracking by conservationists and researchers for whom costs were previously too high, especially, in the Global South. However, further work is necessary to unlock this potential. For example, translation of resources and/or more provision of training could increase SnapperGPS' potential in this regard.

In general, the substantial interest in the SnapperGPS system—including several people who spent time and money on independently replicating the hardware—is evidence for the demand for a system that addresses hypotheses 1–5 and hints at a lack of alternative solutions that would do so.

6

A Low-Cost Snapshot GNSS Receiver with Cloud-Offloading via a Narrow-Band Cellular Connection

6.1 Problem Statement

The SnapperGPS snapshot GNSS system presented in Chapter 4 has weaknesses that limit the variety of use cases it can be employed for, both in wildlife tracking and beyond. One such weakness is the fact that it is based on a pure data logger that mandates recovery of the hardware to access any data. This excludes its use for two wildlife tracking scenarios:

- A scenario where device recovery is challenging because of a high prior uncertainty of the expected movement of the tagged animal [127]. Reliable tag recovery is only possible if an approximate location of the animal at a certain time range in the future can be predicted with sufficient accuracy [128], for example, in case of nesting reptiles and birds, animals with established drinking or feeding places, species with small habitats [6] (including many reptiles and small mammals), and migrating animals with known way points.
- A scenario in which data is not only needed to study animal behaviour in hindsight, but also in near-real time, e.g., to protect an individual animal

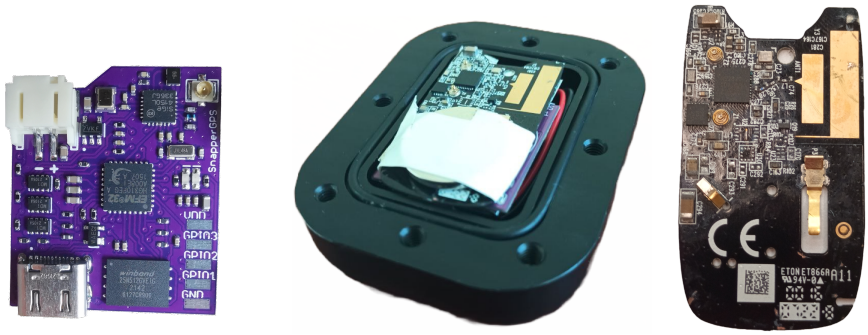


Figure 6.1: SnapperGPS receiver (left) with a Bluetooth beacon (right) [132] to aid device recovery. The range of this beacon is about 120 m (according to its datasheet and verified in tests), which is much shorter than that of a VHF transmitter. However, it can run more than a year on a CR2032 lithium coin battery.

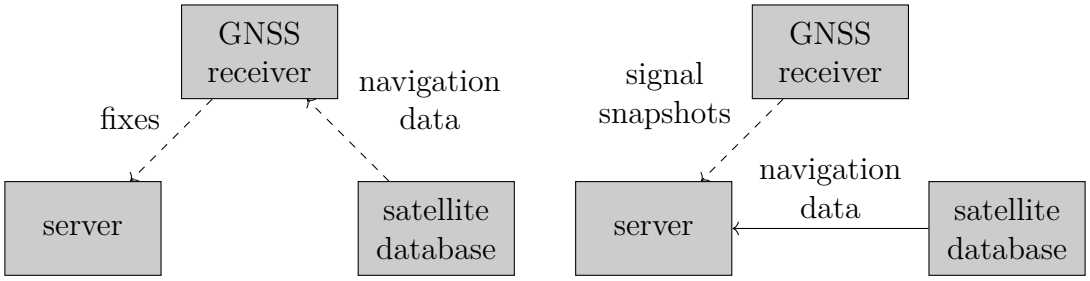
from a current threat, to reduce human-wildlife conflicts, or to monitor animal health and welfare [127, 129–131].

A solution approach to the first scenario is to co-deploy a secondary localisation technology, such as a VHF transmitter or a Bluetooth beacon, see Figure 6.1. It is inactive during the deployment to not jeopardise the low-power benefits of the snapshot technology and only transmits at a pre-configured end time of the deployment to aid recovery.

An alternative approach that addresses both scenarios is to connect the snapshot GNSS receiver to a wireless network [6, 47, 52, 133, 134]. This enables online data transmission to the cloud with the snapshot processing back-end. The topology of this approach is similar to the one of A-GNSS (Chapter 2.10), but while A-GNSS requires the down-link of a network connection, a pure snapshot GNSS receiver only uses the up-link, see Figure 6.2.

Traditional GNSS receivers paired with wireless transmitters to relay position fixes that are calculated on board are already used for many applications. Recent publications propose the use of narrow-band cellular networks. They offer good spatial coverage and high data rates in contrast to low-power wide-area networks (LPWANs) and low power consumptions in contrast to other cellular networks [135–137].

However, the European GNSS Agency (GSA) predicts that a network-connected pure snapshot GNSS receiver that transmits raw GNSS snapshots can theoretically



(a) Traditional assisted GNSS (A-GNSS). **(b)** Snapshot GNSS.

Figure 6.2: Comparison of traditional assisted GNSS (A-GNSS) and snapshot GNSS with potentially wireless connections as dashed arrows. The goal is to have position fixes available to an end-user on a server (bottom-left). In both cases, satellite navigation data from a public database (bottom-right) is used to reduce the GNSS signal acquisition time of the receiver. However, the information flow differs with the navigation data either being sent to the receiver or to the server for processing.

realise energy savings of a factor of 15 to 25 in contrast to traditional GNSS, even if large 20 KB snapshots are sent [47].

Implementations are rare, though. u-blox offers a commercial cloud-based snapshot GNSS solution [133, 134]. But it is unlikely that this is a pure snapshot GNSS implementation. Furthermore, it is not supported by dedicated snapshot-only receivers, only by complete GNSS receivers with on-board processing capabilities. This increases hardware complexity in contrast to a receiver that is only designed to capture signal snapshots, but not to execute processing. According to the datasheet, the approach still reduces the energy consumption of the GNSS receiver to 10% of the overall device with the cellular modem (LTE-M) for wireless connectivity accounting for the remaining 90%. This is in contrast to traditional setups that usually have energy budget ratios of at least 50:50 due to the comparably high energy consumption of traditional GNSS signal acquisition, tracking, and on-board processing, cf. Section 2. The u-blox solution is not directly available to end-users, though.

Wang et al. present a pure network-based snapshot GNSS system where only raw GNSS snapshots are transmitted [52]. However, it has limited applicability in practise because it uses LoRaWAN as wireless network technology. LoRaWAN either requires setting up network infrastructure in an area where the system shall be deployed or using a public LoRaWAN network such as The Things Network.

However, The Things Network has limited coverage and its up-link airtime is capped to 30 s per day per node at a maximum bit rate of 11 kbit/s (EU) / 21.9 kbit/s (USA) [138]. This does only allow for the transmission of two to five of their 15.5 KB GNSS snapshots per day under ideal conditions and less considering that data rates further from an access point are much lower. Finally, The Things Network's maximum payload size is 222 bit (EU) / 242 bit (US) [138], requiring splitting of a single snapshot across many individual messages.

Considering the aforementioned limitations, the question that guides this chapter shall be if the snapshot GNSS technology can be used to build a low-cost, low-power, and easily deployable localisation system, where data is transmitted and computed locations are available to a remote operator before physical recovery of the deployed node. Such a system could potentially be useful not only for animal tracking but also for asset and logistics tracking. (Note that recovery of the deployed note is usually still important in animal tracking scenarios, even if remote data download is possible. This is for animal welfare reasons.)

6.2 System Design

This section presents the design of a complete wireless snapshot GNSS system starting with the choice of the network technology (Section 6.2.1) followed by the hardware setup (Section 6.2.2) and the software (Section 6.2.3).

6.2.1 Wireless Technology

The choice of the network technology is crucial for the overall performance of the system. Data transmission by satellite up-link has traditionally been used for wildlife tracking due to its global coverage, cf. Section 2.14. However, its low data rates, high receiver costs, high data costs, and high energy consumption prohibits its use for a low-cost, low-power snapshot GNSS receiver, which needs to transmit several kilobytes of raw data per location rather than a few bytes for latitude and longitude of a fix computed on-board. Potentially applicable ground-based wireless networks include low-power wide-area networks (LPWANs) such as LoRaWAN,

narrow-band cellular networks designed for the internet of things (IoT) such as LTE-M (Cat M1, eMTC) or NB-IoT (Cat NB1, Cat NB2), and regular broad-band cellular networks such as 4G and 5G.

LoRa is the physical layer used in LoRaWAN. It is designed for devices that operate for multiple years on battery at the expense of low maximum data rates between 0.3 and 50 kbps [139]. This would require powering a device for many seconds to transmit raw GNSS snapshots with 6 KB each, especially, if the node is far from the base station, and hence transmission rates are significantly lower than the achievable maximums. In consequence, it is only a valid solution for snapshot GNSS in scenarios where a low location sampling rate is acceptable. Another drawback is the limited coverage of existing LoRa networks, especially outside of Europe and outside of urban areas. Thus, for wildlife tracking applications, it would often be required to set up a dedicated network infrastructure with a base station providing coverage in a radius of 2–5 km in urban areas and up to 15 km in sub-urban areas [139]. This issue is even more significant for logistics and asset tracking, where coverage would be required along a potentially long route spanning multiple countries.

In contrast, popular broad-band cellular networks have high coverage and low costs. (The average price for 1 GB is below $3 in the USA [140] and 1 GB allows the transmission of more than 100,000 snapshots captured by a SnapperGPS receiver.) Even for remote wildlife tracking scenarios, it can often be expected that tagged animals will at least sporadically visit areas with cellular coverage, which would allow batch uploads of GNSS snapshots from time to time. However, 4G and 5G come with unnecessarily high upload data rates of up to 150 Mbit/s and 1 Gbit/s, respectively, at the expense of increased energy consumption. Older technologies such as 3G and 2G are being phased out [141].

A spot in between is occupied by narrow-band cellular networks such as NB-IoT and LTE-M, which are often co-deployed with 4G and 5G (cf. Figure 6.3), but whose lower bandwidth reduces power consumption. Maximum up-link data rates of 1119 kbit/s (LTE-M Cat M1), 158.5 kbit/s (NB-IoT Cat NB2), or 70 kbit/s (NB-IoT

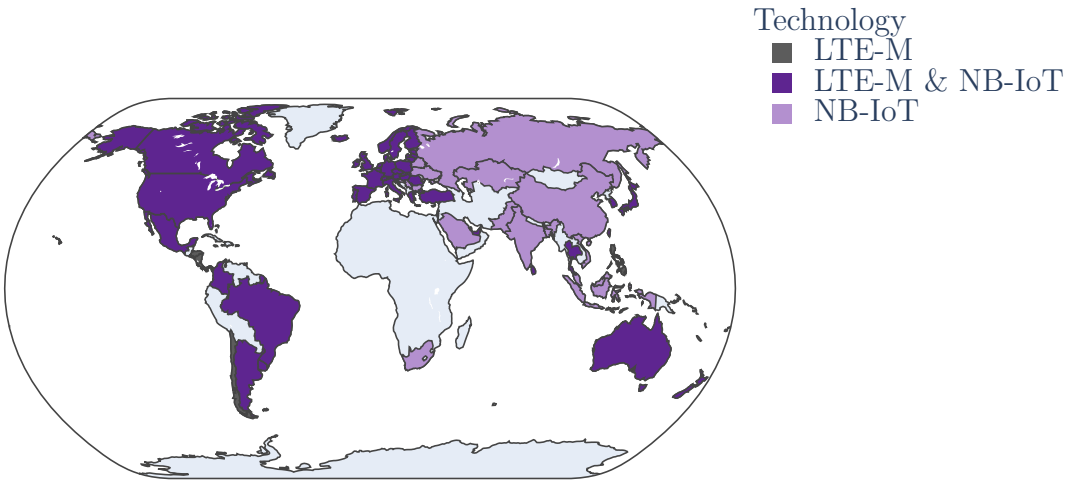


Figure 6.3: Map of countries and territories that have deployed narrow-band cellular networks for mobile IoT (May 2023). Created using data from the GSM Association [145]. Commercial deployments have started in 2017 and are expected to increase further in the future [144].

Cat NB1), respectively, still render uploading large number of snapshots feasible. LTE-M has a narrow bandwidth of 1.4 MHz compared to 20 MHz for regular LTE, giving a longer range but less throughput [142, 143]. It can be used for secure and end-to-end communication and supports cell handover as well as roaming, which both are advantageous for mobile tacking applications [142–144]. NB-IoT does not support cell handover, but has a bandwidth of 200 kHz and hence an even longer range and better penetration, but lower throughput compared to LTE-M and regular LTE. It is especially useful for low-power static applications [142, 143].

In consequence, network connectivity via a narrow-band cellular network will be evaluated in the following, using LTE-M in particular for this mobile rather than static application. Specifically, the LTE-M standard that is currently (2023) deployed, eMTC (enhanced Machine Type Communication), also known as LTE Cat M1.

6.2.2 Electronics

The hardware setup shown in Figure 6.4 is based on a regular SnapperGPS receiver, in particular version V1.0.0 using an external active patch antenna. The cellular module is a Blues Wireless LTE-M Notecard [146], which is a cellular

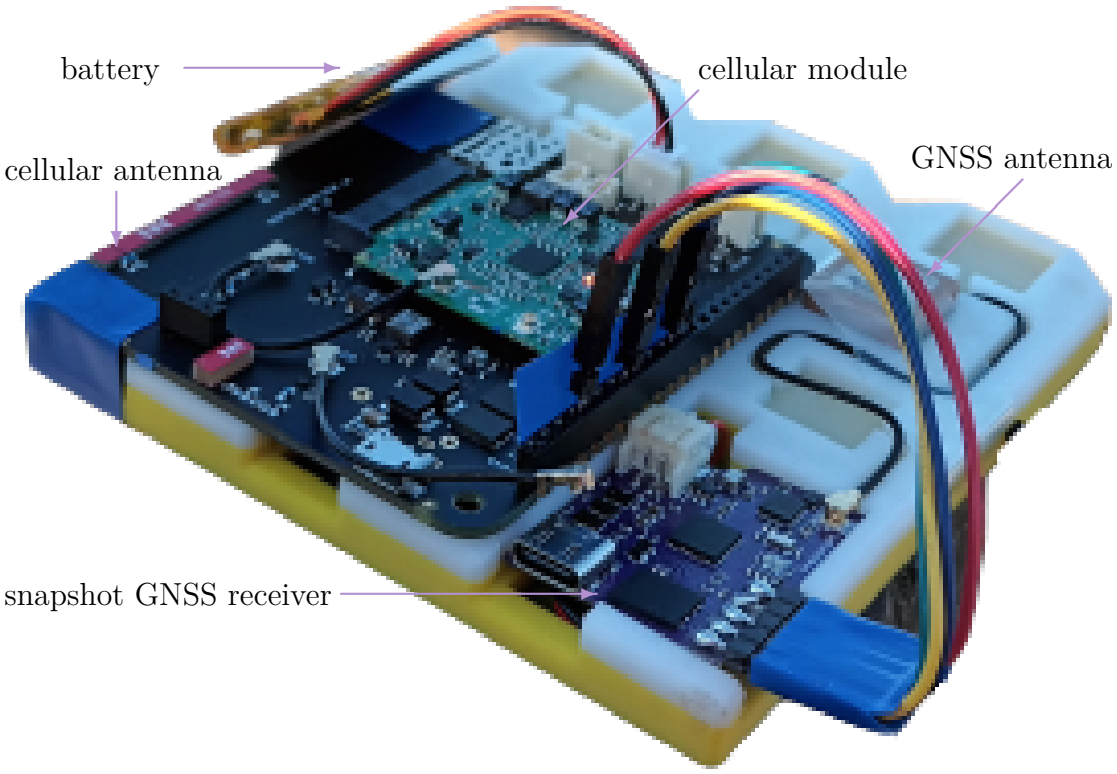


Figure 6.4: Test setup for a snapshot GNSS receiver with wireless cloud-offloading.

IoT module build around a Quectel BG95-M3 cellular modem [142]. Both boards are connected through a Blues Wireless Notecarrier A [147]. The latter connects to a LiPo battery that supplies all boards with power. The Notecarrier also has a cellular antenna and exposes the I2C interface of the Notecard, which connects to the I2C interface exposed by the accessible solder pads of the SnapperGPS board. An advantage of the Notecarrier is that it is open source. In consequence, its relevant components could easily be integrated into a slightly extended SnapperGPS board, theoretically reducing the overall setup to this modified SnapperGPS board, the similarly sized Notecard, a GNSS antenna, and the battery[1]. The Notecard itself has a few properties that render it suitable for the envisioned prototype, including a low energy consumption (about 10 J for a synchronisation, which is about 0.8 mA h at 3.3 V) and that it comes with network connectivity in more than 135 countries and 500 MB of data at a comparably low price for hardware and data combined

[1]It was not possible to actually build this integrated setup due to the unavailability of ICs caused by the global chip shortage since 2020, which is still on-going as of writing (2023).

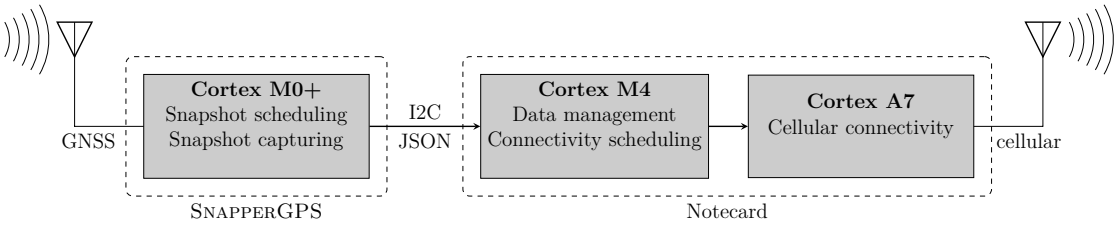


Figure 6.5: The three MCUs and two antennas of the wireless SnapperGPS setup, including their individual tasks. The MCU on the left uses the least power and is the most frequently active one. The MCU on the right uses the most power and is the least frequently active one.

of $59, roughly the same magnitude as the price of a SnapperGPS receiver, cf. Section 4.2.8. In theory, 500 MB of payload data allow for the transmission of more than 80,000 snapshots, which would allow for a fix every 30 min for five years, for example. Additional 500 MB can be purchased for $15.

6.2.3 Software

The whole setup includes three MCUs: the SnapperGPS's EFM32 Happy Gecko with a Cortex M0+ core acts as host and the cellular IoT module with a Cortex A7 and a Cortex M4 as client. This allows to use the low-power Cortex M0+ for the frequent snapshot capture, the M4 with slightly higher energy consumption to receive the data from the SnapperGPS board via I2C, data management, and scheduling of network connectivity, and finally the more powerful A7 for infrequent cellular connectivity only, see Figure 6.5.

The SnapperGPS board transfers data to the cellular IoT module in the JSON format via I2C using custom firmware on the former and the default Notecard firmware on the latter.

The cellular module is configured to attempt a batch upload of GNSS snapshots via LTE-M (or NB-IoT) in regular intervals. However, the cellular module also operates a so-called *penalty box*, which increases upload time intervals after repeated network registration failures or in case of low power supply. This is to prevent unnecessarily draining the battery.

In general, for many applications, it is not necessary to transmit all collected GNSS snapshots. Often, only the current location is relevant to monitor the

location and/or to aid with device recovery. Then, the full amount of data can be downloaded after recovery via a wired connection. In consequence, the transmission of more recently captured snapshots is prioritised when transmitting data to the cloud (last-in-first-out).

The uploaded snapshots are first routed to an intermediate server (Blues Wireless Notehub) and then pulled by a modified version of the SnapperGPS back-end (Section 3.4) for processing. The web app presents the resulting fixes to an end-user[2].

6.3 Evaluation

Since the localisation accuracy of SnapperGPS has already been evaluated in Section 3.3 and Section 4.5.1, the focus of the following evaluation is on the energy consumption of the proposed system as well as the comparison with competing implementations. The latter includes:

- An experimental comparison with a baseline system: a similar setup with an LTE-M Notecard and a Notecarrier, which uses a traditional GNSS receiver that calculates fixes on board. It only relays position fixes to the cloud. Data transmission is done in the same way as by the wireless SnapperGPS setup.
- A theoretical comparison with a comparable A-GNSS system.

6.3.1 Experiments

Three types of charge consumption measurements were recorded for different locations in Oxford, UK (cf. Figure 6.6) at different times across several days:

- The energy that the proposed system consisting of SnapperGPS board and cellular module needs to upload a certain number of SnapperGPS snapshots via LTE-M to the cloud,

[2]If one the cellular SnapperGPS prototypes happened to be online during the last hour, then you can track it on `https://snappergps.info/live`.

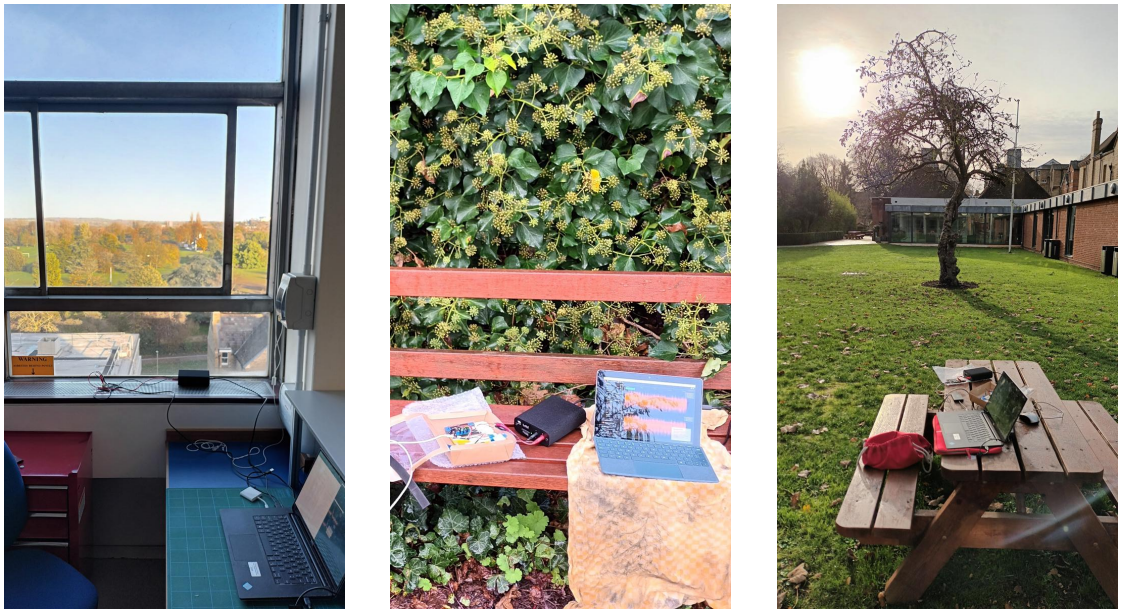


Figure 6.6: Left: test setup indoors next to a window with limited sky visibility. Centre: test setup outdoors next to a hedge with partial sky visibility. Right: test setup outdoors with good sky visibility. The energy consumption for capturing a GNSS signal snapshot is identical in all three scenarios, while the energy consumption for attempting an on-board fix decreases from the left to the right.

- The energy that the baseline system consisting of on-board GNSS module and cellular module needs to acquire a GNSS fix, and
- The energy that the cellular module of the baseline system needs to upload a certain number of GNSS fixes.

In addition, the sleeping current consumption of both setups when no data capturing or transmission is occurring is measured. The final ingredient for the comparison of the two systems is the energy needed by the SnapperGPS board to capture a snapshot, but this is independent from time and location and can be taken from Section 4.5.2.

For all measurements, an external portable power supply with in-build power monitoring powers the setup rather than a 3.7 V LiPo battery.

6.3.2 Results

Figure 6.7 presents the measurements for the snapshot-based system: An unsuccessful snapshot upload attempt (zero snapshots uploaded), e.g., because no cellular

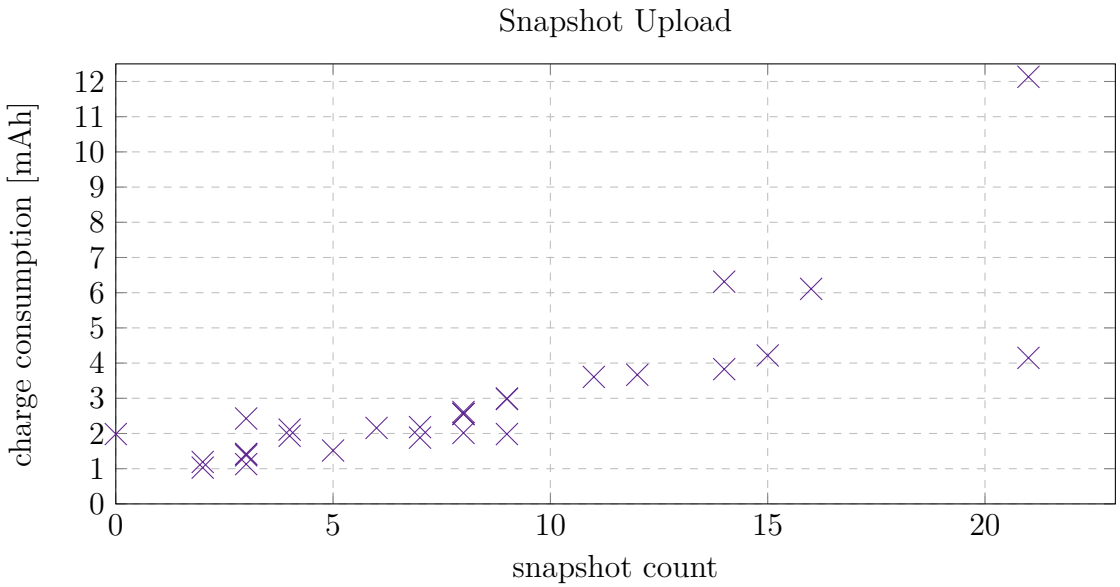
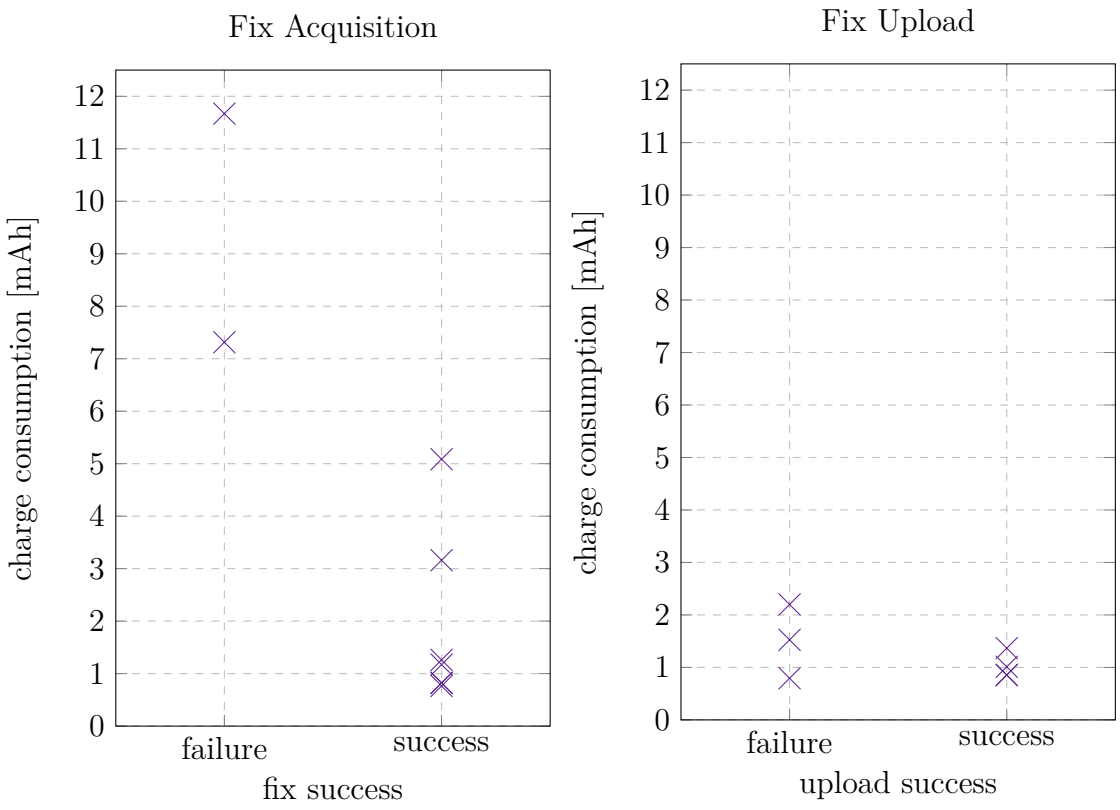


Figure 6.7: Charge consumption per session over number of GNSS signal snapshots uploaded per session ($n = 29$).

tower is available, costs about 2 mA h. Uploading roughly 1–10 snapshots costs about 1–3 mA h. The transmission overhead dominates the energy consumption in this region, not the size of the actual payload. This is in line with Blues Wireless stating that payload size only becomes relevant if it is larger than around 100 KB. For roughly 10–20 snapshots, the charge consumption slightly increases to around 4 mA h, with a few outliers when network connectivity is bad. In both cases, there is some variability depending on the location and the time of the day. SnapperGPS's charge consumption when acquiring a snapshot is <0.3 µA h and, hence, negligible in contrast to the energy needed for an upload. It is omitted in the following.

Figure 6.8a shows the energy expenses for acquiring a fix with the on-board GNSS module employed for comparison. It significantly depends on the amount of sky visibility. With good sky visibility, it runs about 30 s and consumes 1 mA h. With partial sky visibility, e.g., outdoors next to buildings or trees, this increases to a few mAh until a fix is achieved. With severely limited sky visibility, e.g., indoors next to a window (Figure 6.6), the on-board GNSS module does not get a fix at all and can run for 900 s until the default time-out is reached. (Notably, SnapperGPS does achieve valid fixes in the same scenarios.) Uploading a number



(a) Charge consumption per on-board GNSS fix attempt, depending on whether the attempt fails or succeeds ($n = 9$).

(b) Charge consumption per session when uploading on-board GNSS fixes, depending on whether the upload fails or succeeds ($n = 7$).

Figure 6.8: Charge consumption for acquiring and uploading on-board fixes.

of GNSS fixes needs 1–2 mA h in total per session. This is almost independent of the number of fixes transmitted due to their small size.

The sleeping current for the Blues Wireless Notecard and Notecarrier used for both, the proposed system as well as the baseline one, is 8 µA. Adding a SnapperGPS consequently causes the current consumption to raise to 9–10 µA.

6.3.3 Discussion

In the examined scenarios, the proposed system (snapshot GNSS plus cellular) outperforms the evaluated baseline system (on-board GNSS plus cellular) with respect to the energy consumption. This is because acquiring and transmitting a snapshot consumes less than 1 mA h per snapshot as soon as the upload batch size is at least two snapshots. In contrast, using the on-board GNSS and the cellular

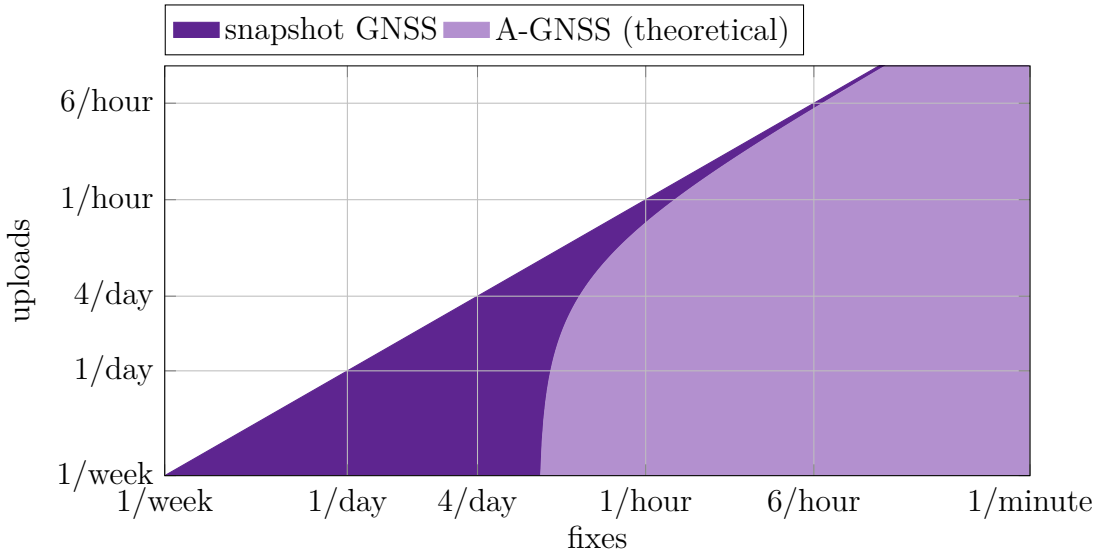


Figure 6.9: Energy consumption comparison of snapshot GNSS and A-GNSS depending on the frequency of fixes and the frequency of uploads. The domain where snapshot GNSS is more energy-efficient is shaded purple. The domain where A-GNSS is more energy-efficient is shaded pink. The values for snapshot GNSS are taken from the experiments while the values for A-GNSS are based on the measurements for the traditional GNSS setup and augmented with the assumption that a daily download of 10 KB of assistance data reduces the time-to-fix by 90%. Failures to acquire a fix as well as upload failures are not considered.

module to acquire and transmit a GNSS fix takes always more than 1 mA h per fix, even for large upload batch sizes. The difference in sleeping current is negligible.

However, it can be questioned if there are no better baselines than the Blues Wireless setup. The long and energy expensive run times of the GNSS module of at least about 30 s could be brought down to a few seconds maximum by using A-GNSS (see Section 2.10 and Figure 6.2), potentially cutting energy consumption by at least 90%. This would drop the charge consumption of the Notecard for acquiring a single on-board fix to about 0.1 mA h in contrast to 1 mA h for a fix upload. This requires regularly downloading navigation data though, which adds energy consumption. However, commercial providers like u-blox or Baseband Technologies offer compressed navigation data at roughly 10 KB per day with data available for download up to one month in advance [62, 63]. This is slightly less than the size of two GNSS signal snapshots at 6 KB each. Therefore, the navigation data download would add about 1 mA h if executed daily. Figure 6.9

shows that as long as fixes are sufficiently more often captured than uploaded, A-GNSS outperforms the energy consumption of the proposed system.

The snapshot GNSS solution only excels in specific scenarios such as:

- Only a few snapshots are uploaded at the end of a deployment to aid device recovery prior to wired download of the remaining data.
- Sometimes only a window of less than a few seconds is available to capture GNSS data, which is too short for traditional A-GNSS, but sufficient for snapshot GNSS.

The costs of the electronics of the proposed wireless snapshot GNSS system are less than $100, but the energy consumption increases significantly in contrast to the data loggers presented in Section 4. It is still possible to upload thousands of snapshots on the charge of three 1.5 V alkaline AA batteries with 2500 mA h or a LiPo battery with similar capacity, but such a device would weigh around 100 g—as much as a chocolate bar—and hence would be much bulkier than a 10 g snapshot GNSS logger, denying its deployment on small animals.

6.4 Conclusions

Adding a narrow-band cellular modem to a snapshot GNSS receiver is a viable option to enable near-real-time monitoring or to aid device recovery.

However, a theoretical A-GNSS solution is likely to outperform a pure snapshot GNSS solution if data capture on at least an hourly basis is desired.

A general drawback is the limited coverage of cellular networks, prohibiting deployments on animals that never enter areas with coverage. However, cellular networks are among the ground-based networks with the best coverage globally.

7

Smoothing Algorithms for High-Rate Snapshot GNSS

7.1 Problem Statement

Chapter 3 discusses algorithms that robustly estimate locations from short, low-quality GNSS signal snapshots. When processing a recording, the algorithms treat the position estimation for each incoming snapshot mostly isolated[1]. The core optimisation problem always considers only input data from a single timestamp rather than previous or subsequent observations. However, both, previous and subsequent observations, are available during offline snapshot processing. Conditioning the current estimate on those is called smoothing (Section 2.12), essentially low-pass filtering the track and reducing high-frequent noise. If the location sampling interval is short enough, that is, less than half of the smallest time constants of the movement patterns of interest, then this procedure should reduce estimation

[1]Exceptions are (cf. Section 3.2):

- Employing the first few snapshots to estimate hardware error parameters that are used for the whole dataset.
- Propagating the receiver clock error between consecutive snapshots.
- Initialising a new estimation with the most recent plausible estimate.
- Centring the search space at the most recent plausible estimate.
- Using consecutive estimates for plausibility checks.

uncertainty. Given this potential, this chapter experimentally explores different approaches to smoothing for snapshot positioning to:

- Assess the achievable localisation accuracy gains,
- Compare the performance of different smoothing algorithms,
- Compare the performance of different movements models,
- Compare the performance of loosely coupled and tightly coupled snapshot positioning and smoothing, using a novel robust approach based on incremental factor graph optimisation to the latter, and
- Establish the range of sampling intervals for which smoothing is beneficial.

To the best knowledge, this is the first study of this kind for snapshot GNSS. Section 7.2 presents the models and algorithms, Section 7.3 shows their evaluation on two new large datasets, and Section 7.4 concludes the chapter with recommendations for the use of smoothing in practice and ideas for future work.

7.2 Models and Algorithms

This chapter considers two types of algorithms; those that separate snapshot positioning and smoothing and those that tightly couple snapshot positioning and smoothing:

- *Loosely coupled smoothing* takes the outputs from the snapshot positioning algorithms from Chapter 3 (location and timestamp estimates) as inputs and outputs refined estimates. It treats (i) snapshot positioning and (ii) smoothing as separate optimisation problems. The smoothing can be abstracted as obtaining fixes from a "global position sensor" and fusing them with a motion model. Using traditional (non-snapshot) GNSS receivers, this is highly popular, for example, in simple robotics applications [69, 72].

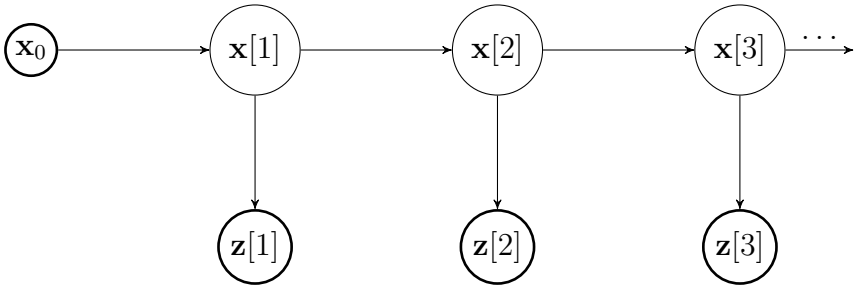


Figure 7.1: Hidden Markov model of the loosely coupled estimation problem represented as Bayesian network. Each node is associated with a conditional probability density: the top Markov chain contains the prior probability $P(\mathbf{x}_0)$ of the state and transition probabilities $P(\mathbf{x}[1]|\mathbf{x}_0)$, $P(\mathbf{x}[2]|\mathbf{x}[1])$, and $P(\mathbf{x}[3]|\mathbf{x}[2])$ between consecutive states (true receiver locations). Observations (observed location fixes) $\mathbf{z}[k]$ depend only on the state $\mathbf{x}[k]$, modelled by conditional probability densities $P(\mathbf{z}[k]|\mathbf{x}[k])$. Given observations $\mathbf{z}[1]$, $\mathbf{z}[2]$, and $\mathbf{z}[3]$, the goal is to estimate the hidden states $\mathbf{x}[1]$, $\mathbf{x}[2]$, and $\mathbf{x}[3]$ that maximise the posterior probability $P(\mathbf{x}[1], \mathbf{x}[2], \mathbf{x}[3]|\mathbf{x}_0, \mathbf{z}[1], \mathbf{z}[2], \mathbf{z}[3])$.

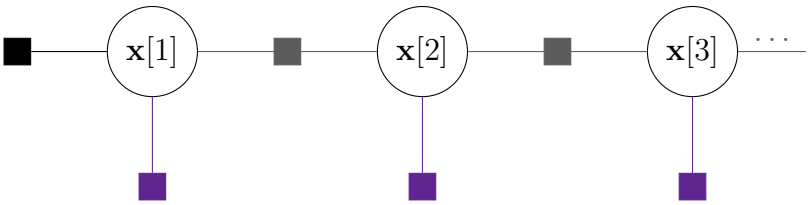


Figure 7.2: Hidden Markov model of the loosely coupled estimation problem represented as factor graph with two types of nodes: variable nodes ◯ for states $\mathbf{x}[k]$ and factor nodes for priors ■, motion models ■, and pre-computed location fixes ■.

- *Tightly coupled smoothing* solves snapshot positioning and smoothing in a single optimisation problem. For traditional GNSS receivers, tight integration of raw GNSS observations when estimating trajectories has been shown to improve robustness and accuracy in robotics applications where accurate DGNSS is not available, cf. Section 2.13.

Section 7.2.1 and Section 7.2.2 cover loosely coupled algorithms as baselines and Section 7.2.3 proposes a tightly coupled one.

7.2.1 Rauch-Tung-Striebel Smoother

Figure 7.1 and Figure 7.2 show the loosely coupled smoothing problem as hidden Markov model (HMM). The states $\mathbf{x}[k] \in \mathbb{R}^p$ contain the unknown (hidden) true 2D locations at time index $k \in \mathbb{N}$. The observations $\mathbf{z}[k] \in \mathbb{R}^q$ are the

locations estimated with snapshot positioning ($q = 2$). For both, local Cartesian easting/northing coordinates on a tangent plane of the Earth are used. The observations are assumed to be conditional independent given the states and the states are assumed to form a Markov chain. For the observations and the state transitions, linear models with additive Gaussian noise are considered. Let $\mathbf{F} \in \mathbb{R}^{p\times p}$ be the state transition matrix, $\mathbf{H} \in \mathbb{R}^{q\times p}$ the measurement matrix, $\mathbf{Q} \in \mathbb{R}^{p\times p}$ the covariance of the process noise $\eta_{\mathrm{x}}[k] \propto \mathcal{N}(\mathbf{0}, \mathbf{Q})$, and $\mathbf{R}[k] \in \mathbb{R}^{q\times q}$ the covariance of the measurement noise $\eta_{\mathrm{z}}[k] \propto \mathcal{N}(\mathbf{0}, \mathbf{R})$. Then the state space model is

$$\begin{aligned} \mathbf{x}[k] &= \mathbf{F} \cdot \mathbf{x}[k-1] + \eta_{\mathrm{x}}[k] \\ \mathbf{z}[k] &= \mathbf{H} \cdot \mathbf{x}[k] + \eta_{\mathrm{z}}[k]. \end{aligned} \tag{7.1}$$

For a 2D state space model based on a constant-position-random-velocity model, the matrices are

$$\mathbf{F} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.2}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \tag{7.3}$$

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^2}{2} & 0 \\ 0 & \frac{\Delta t^2}{2} \end{bmatrix} \sigma_{\mathrm{Q}}^2 \tag{7.4}$$

$$\mathbf{R}[k] = \begin{bmatrix} \frac{\sigma_{\mathrm{z}}[k]}{\sqrt{2}} & 0 \\ 0 & \frac{\sigma_{\mathrm{z}}[k]}{\sqrt{2}} \end{bmatrix}, \tag{7.5}$$

where $\Delta t \in \mathbb{R}^+$ is the snapshot interval, $\sigma_{\mathrm{Q}} \in \mathbb{R}^+$ a process noise scaling factor, and $\sigma_{\mathrm{z}}[k] \in \mathbb{R}^+$ the confidence radius provided by the snapshot positioning algorithm for fix $\mathbf{z}[k]$. An alternative 4D state-space model with the 2D position and 2D velocity as state and a constant-velocity-random-acceleration model is [148]

$$\mathbf{F} = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{7.6}$$

$$\mathbf{H} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{7.7}$$

$$\mathbf{Q} = \begin{bmatrix} \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} & 0 \\ 0 & \frac{\Delta t^4}{4} & 0 & \frac{\Delta t^3}{2} \\ \frac{\Delta t^3}{2} & 0 & \Delta t^2 & 0 \\ 0 & \frac{\Delta t^3}{2} & 0 & \Delta t^2 \end{bmatrix} \sigma_{\mathrm{Q}}^2 \tag{7.8}$$

$$\mathbf{R}[k] = \begin{bmatrix} \frac{\sigma_{\mathrm{z}}[k]}{\sqrt{2}} & 0 \\ 0 & \frac{\sigma_{\mathrm{z}}[k]}{\sqrt{2}} \end{bmatrix}. \tag{7.9}$$

For both models, the Kalman filter is the optimal causal filter. The Rauch-Tung-Striebel (RTS) smoother is the equivalent smoother to the Kalman filter, solving the problem

$$\hat{\mathbf{x}}[k] = \underset{\mathbf{x}[k]}{\operatorname{argmax}} \quad P\left(\mathbf{x}[k] | \mathbf{x}_0, \mathbf{z}[1], \ldots, \mathbf{z}[n]\right) \quad \forall k \in 1 \ldots n. \tag{7.10}$$

The RTS smoother consists of a Kalman filter forward pass and an analogue recursive backward pass.

Advantages of the RTS smoother are its linear time complexity $O(n)$ w.r.t. the number of observations $n \in \mathbb{N}^+$ and a low runtime in general due to the existence of a closed-form solution. On the other hand, the underlying assumption of Gaussian measurement and process noise is a limitation. If the input data contains outliers, the RTS smoother is significantly affected. Therefore, outliers must be removed before it is applied. This is done by removing observations for which the snapshot positioning algorithm estimates a high uncertainty and requiring the RTS smoother to perform interpolation for those timestamps.

Theoretically, it is possible to solve the estimation problem (7.10) with another optimisation method such as Markov chain Monte Carlo (MCMC) methods or factor graph optimisation (FGO) operating on a Bayes tree, cf. Section 2.12. However, no performance improvements are expected since the underlying models are identical and linear.

While the considered linear models aim at generalising over different application domains, for individual species, it would be reasonable to use more complex, potentially non-linear, movement models (state transitions), incorporating, e.g., wind speeds for birds [149] or currents for aquatic species.

7.2.2 Gaussian Process Regression

An alternative to the state-space representation is to model the 2D locations obtained with snapshot positioning as Gaussian processes (GPs)

$$x_i[k] \propto \mathcal{GP}\left(m_i\left(k\right), \phi_i\left(k, k'\right)\right) \quad \forall i \in \{1, 2\}, \tag{7.11}$$

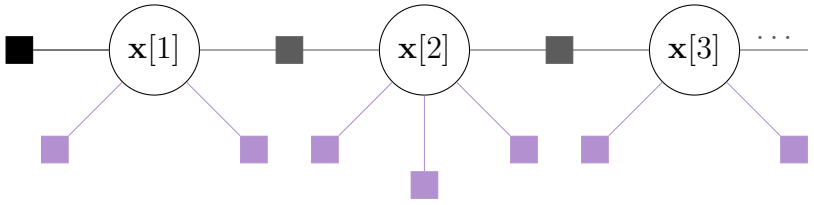


Figure 7.3: Hidden Markov model of the tightly coupled estimation problem represented as factor graph with two types of nodes: variable nodes ◯ for states $\mathbf{x}[k]$ and factor nodes for priors ■, motion models ■, and pseudoranges ■.

where $x_0, x_1 \in \mathbb{R}$ are easting and northing, respectively, $k, k' \in \mathbb{N}$ are time indices, $m_i \colon \mathbb{N} \to \mathbb{R}$ is the mean function and $\phi_i \colon \mathbb{N} \times \mathbb{N} \to \mathbb{R}$ the covariance (kernel) function. For the mean function, the empirical mean is used and for the kernel function a product of a constant and a Matérn kernel

$$\begin{aligned} m_i(k) &= \frac{1}{n} \sum_{k' \in \{1,\dots,n\}} z_i[k'] \\ \phi_i(k, k') &= \sigma^2_{\phi,i} \cdot \phi_{\text{Matérn}}(k, k'; \nu_i, l_i). \end{aligned} \tag{7.12}$$

The Matérn kernel $\phi_{\text{Matérn}}$ is parameterised by the smoothness parameter $\nu_i \in \mathbb{R}^+$ and the length scale $l_i \in \mathbb{R}^+$. It can capture both, short-range and long-range dependencies in the data. The parameters are fitted using maximum likelihood estimation (MLE).

Gaussian noise is assumed on the observations (snapshot positions):

$$\begin{aligned} z_i[k] &= x_i[k] + \eta_{z_i}[k] \\ \eta_{z_i}[k] &\propto \mathcal{N}\left(0, \frac{\sigma_z[k]}{\sqrt{2}}\right). \end{aligned} \tag{7.13}$$

Based on this model, Gaussian process regression (GPR) is employed to estimate the states $\mathbf{x}[k] \quad \forall k \in \{1, \dots, n\}$ [150].

7.2.3 Tightly Coupled Factor Graph Optimisation

Both, the state-space model in Section 7.2.1 and the GP model in Section 7.2.2, treat location estimates as observations (inputs). Instead, Figure 7.3 shows an alternative model where observations are individual pseudoranges. For all potentially visible satellites $\{1, \dots, N[k]\}$, the pseudorange of satellite $s \in \{1, \dots, N[k]\}$ is modelled according to Equation (2.1) as Gaussian distributed $\rho_s[k] \propto \mathcal{N}(\hat{\rho}_s[k], \sigma_{\rho_s}[k])$. The

state $\mathbf{x}[k] \in \mathbb{R}^5$ is five-dimensional in this case, unlike when FGO is used for traditional GNSS positioning [72, 76–81] (Section 2.13). It consists of the 3D Earth-centred-Earth-fixed (ECEF) receiver position, the common bias[2], and the coarse-time error, as in Chapter 3.

Due to the presence of the pseudoranges, the estimation problem is highly non-linear (see Section 2.9) and the Kalman filter and the RTS smoother cannot be used anymore. An alternative would be to employ the extended Kalman filter (EKF) or its smoothing equivalent. This linearises the model around the best estimate of the mean and covariance. However, it is already known for single-point coarse-time navigation (CTN, Section 2.5) that a single linearisation step is not necessarily sufficient to obtain accurate results for the highly non-linear optimisation problem. Therefore, factor graph optimisation (FGO) has become of interest for the tight integration of raw GNSS data [78] since it (i) runs multiple iterations for each estimate by design and (ii) does not marginalise the previous state immediately, but keeps a history of states and observations over a past window $]k - \Delta k, k]$. The latter allows to keep re-linearising the model around all states in the time window, thus reducing linearisation errors for states that would have already been marginalised by an EKF [72].

An alternative to FGO to solve the highly non-linear optimisation problem would be to employ a particle filter, but an initial implementation revealed a worse runtime-accuracy trade-off in contrast to FGO, in line with literature on localisation [73].

However, existing FGO algorithms for tight fusion of pseudoranges (Section 2.13) cannot be directly applied to the low-cost snapshot GNSS problem by simply adding an additional state (the coarse time) and replacing the pseudorange observation model with a code-phase observation model. As in Chapter 3, outliers in the observed code phases and, hence, reconstructed pseudoranges, pose a challenge since they are not captured by the Gaussian noise model, but occur frequently due to the use of low-cost hardware, the weakness of the GNSS signals, and multi-path signal receptions. Therefore, the FGO is not solved with a single batch optimisation over all

[2]A single common bias is used for all GNSS because the error introduced by this is expected to be significantly smaller than the error resulting from the imprecisely reconstructed pseudoranges.

Algorithm 4 Tightly coupled snapshot positioning and smoothing via incremental factor graph optimisation

Inputs: Code phases and SNRs for all satellites $s \in \{1, \ldots, N[k]\}$ and samples $k \in \{1, \ldots, n\}$, coarse times for all samples $k \in \{1, \ldots, n\}$, and initial position
Outputs: State estimates $\hat{\mathbf{x}}[k]$ for all samples $k \in \{1, \ldots, n\}$ and uncertainties

for $k \in \{1, \ldots, n\}$ **do**
  Select initial set $S[k]$ of satellites with the highest prior probability of being reliable given the SNRs, cf. Section 3.2.2.
  while estimate $\hat{\mathbf{x}}[k]$ is not plausible **do**
    Predict pseudoranges $\hat{\rho}_s[k]$ from code phases for all satellites $s \in S[k]$.
    FGO over time window Δk to solve $\hat{\mathbf{x}}[k-\Delta k+1] \ldots \hat{\mathbf{x}}[k]$
    $= \underset{\mathbf{x}[k-\Delta k+1] \ldots \mathbf{x}[k]}{\operatorname{argmax}} P(\mathbf{x}[k-\Delta k+1] \ldots \mathbf{x}[k] | \mathbf{x}[k-\Delta k], \rho_{s \in S[k-\Delta k+1]} \cdots \rho_{s \in S[k]})$.
    if estimate $\hat{\mathbf{x}}[k]$ is not plausible **then**
      Remove a satellite from set $S[k]$ using rule from Section 3.2.2.
    end if
  end while
end for
return estimates $\hat{\mathbf{x}}[k]$ for all samples $k \in \{1, \ldots, n\}$ and uncertainties

samples $1, \ldots, n$, but incrementally over a time window $\Delta k \in \mathbb{N}$ using Algorithm 4. It optimises not only the state $\mathbf{x}[k]$, but also the set $S[k] \in \mathcal{P}(\{1, \ldots, N[k]\})$ of satellites to use among the available satellites $\{1, \ldots, N[k]\}$. For the implementation, a fixed-lag smoother based on the efficient incremental optimiser iSAM2 [71] and the GTSAM factor graph library [69] are employed.

7.3 Evaluation

This section evaluates the performances of the algorithms proposed in Section 7.2 with respect to the points of interest in the problem statement (Section 7.1). Section 7.3.1 introduces the datasets, Section 7.3.2 describes the conducted experiments, and Section 7.3.3 presents the results, which Section 7.3.4 discusses.

7.3.1 Data

Two new dynamic datasets were collected[3][4]. These were created by carrying SnapperGPS receivers while walking or cycling in Oxfordshire, UK. Trials on animals were not considered for the quantitative analysis to simplify ground truth collection and to avoid any unnecessary impact on animal welfare.

- *Walking*: 15 individual recordings/tracks with 26,423 datapoints in total captured in 1 s intervals totalling 7:20 h. Motion during these recordings was slower on average, but acceleration, deceleration, and directional changes were more frequent and usually sudden.

- *Cycling*: Twelve individual recordings/tracks with 27,237 datapoints in total captured in 1 s intervals totalling 7:34 h. Motion during these recordings was on average faster, but acceleration, deceleration, and directional changes were less frequent and usually smooth.

SnapperGPS V1.0.0 boards were used for 18 recordings and V2.0.0 for nine recordings. The more expensive Siretta Echo 27 active patch antenna that was already used for data collection in Section 3.3.1 was only used for six of these recordings. In addition, either the low-cost Abracon APAM2764YK0175 active patch antenna (twelve recordings) or low-cost Taoglas GP.1575.25.2.A.02 (five recordings) or GP.1575.25.4.A.02 (four recordings) passive patch antennas were used.

The datasets are expected to be more challenging than the data collection in Section 3.3.1. Firstly, due to the use of lower-cost antennas and more track sections in forests and along tree-lined avenues. The latter is likely to reduce the number of visible satellites and the signal strengths of visible ones.

Ground truth was collected with real-time kinematic (RTK) receivers (u-blox ZED-F9P) with persistent cellular connections to a base station to receive assistance data and perform differential GNSS for improved accuracy, cf. Section 2.11.

[3] https://github.com/JonasBchrt/snapshot-gnss-data-2
[4] http://dx.doi.org/10.5287/ora-xq5b8xva7

7.3.2 Experiments

All algorithms are separately applied to the two datasets since they contain different motion patterns, as described in the preceding Section 7.3.1.

In total, the localisation accuracies of six algorithms are compared: the individual position fixes from the SnapperGPS algorithms described in Chapter 3, those fixes post-processed by either a two-dimensional RTS smoother, a four-dimensional RTS smoother, a two-dimensional GPR, or a two-dimensional FGO, or, alternatively, the snapshot positioning and smoothing tightly coupled in a single FGO. Algorithm parameters are kept constant across all recordings in a dataset, except for GPR, where hyperparameters are fitted with MLE, cf. Section 7.2.2.

To assess the effects of different sampling intervals, the algorithms are applied to the original 1 Hz datasets as well as the same datasets down-sampled by factor ten and 60. A 1 s interval allows operating a SnapperGPS receiver a little more than 6 h, a 10 s interval a little more than 2.5 days, and a 60 s interval a little more than 15 days.

7.3.3 Results

Algorithm	Walking			Cycling		
	1 s	10 s	60 s	1 s	10 s	60 s
No smoothing	15.4 m	15.4 m	14.5 m	14.5 m	14.5 m	14.6 m
2D RTS smoother	8.2 m	10.5 m	13.5 m	7.7 m	11.8 m	14.5 m
4D RTS smoother	8.1 m	10.2 m	13.7 m	7.4 m	11.3 m	14.6 m
2D GPR	7.9 m	11.0 m	14.7 m	7.5 m	12.0 m	19.8 m
2D FGO	8.1 m	10.4 m	13.6 m	7.5 m	11.6 m	14.6 m
Tight FGO	7.9 m	10.0 m	13.5 m	7.3 m	11.3 m	14.6 m

Table 7.1: Median horizontal localisation errors of different smoothing algorithms considering different sampling intervals and travel modes. (RTS smoother: Rauch-Tung-Striebel smoother. GPR: Gaussian process regression. FGO: factor graph optimisation.) Two-dimensional (2D) smoothers and tight FGO use a constant-position-random-velocity model, except for GPR, which uses adaptive models for the two spatial dimensions. The 4D smoother uses a constant-velocity-random-acceleration model. All estimations loosely coupled (snapshot positioning followed by smoothing) except for tight FGO. Localisation errors without smoothing for comparison.

Figure 7.4 presents estimated tracks for three exemplary recordings and Table 7.1 shows the median horizontal localisation errors w.r.t. ground truth for

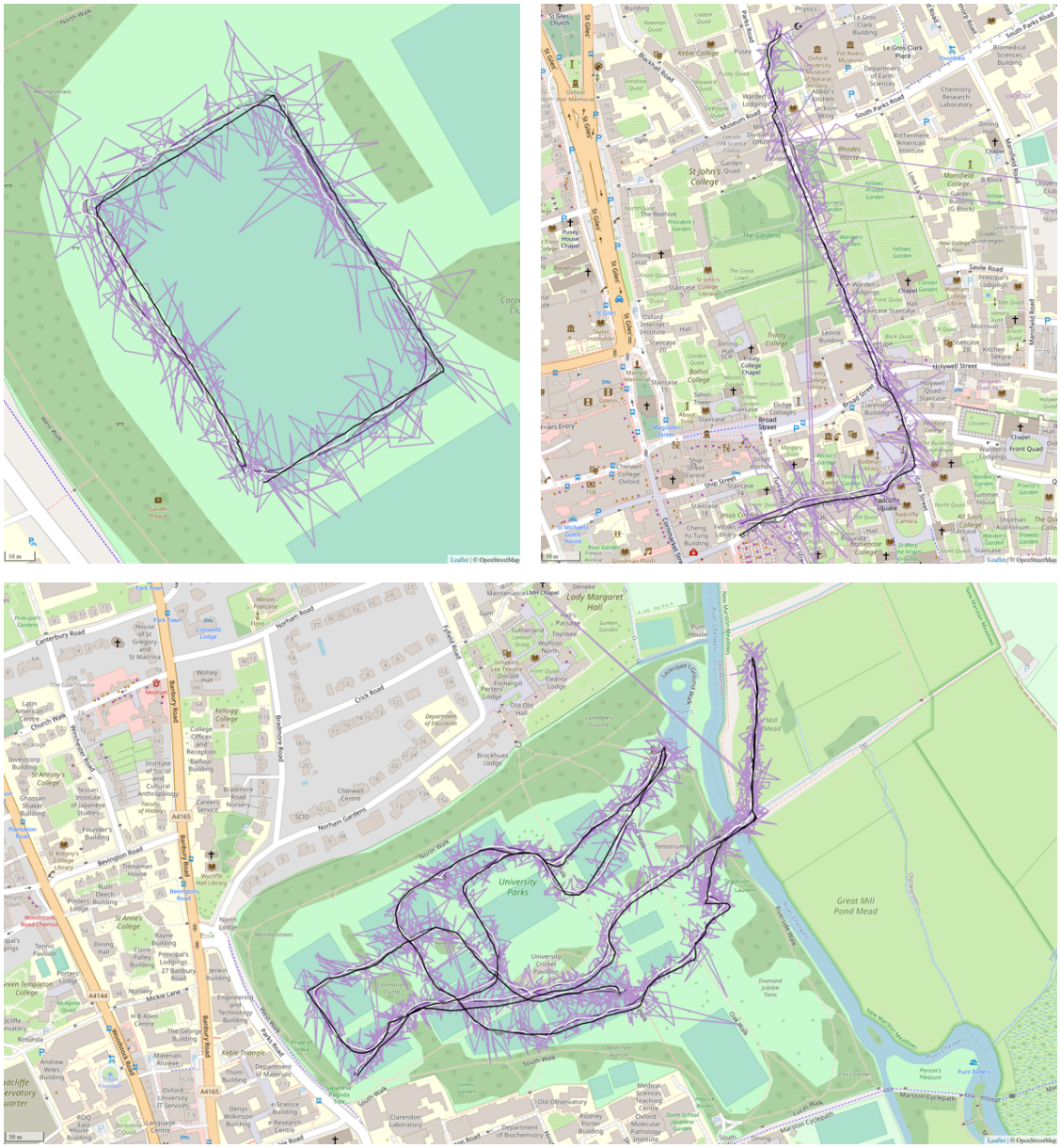


Figure 7.4: Non-smoothed SnapperGPS fixes (pink) from three recordings, 2D RTS smoothing (dark gray), 2D GPR (purple), 2D loosely coupled FGO (light gray), tightly coupled FGO (white), and ground truth (black). **Top-left:** A simple dataset. walking around an American football pitch. Relatively good sky visibility, although the pitch is surrounded by trees on three sides. The median error of the non-smoothed SnapperGPS fixes decreases from 9.8 m to 2.7 m when applying a 2D RTS smoother. **Top-right:** A semi-challenging dataset. Walk along pavements of a city centre with buildings on both sides of the roads, blocking the sky view on the side of the pavement as well as the lower elevation satellites on the opposite side. In addition, the first half is a tree-lined avenue, reducing the satellite signal strength and likely introducing multi-path signals. The median error of the non-smoothed SnapperGPS fixes decreases from 13.3 m to 4.5 m when using the tight FGO. However, no algorithm captures the abrupt turn at the bottom-right. **Bottom:** A simpler dataset. Walking through an urban park with sections with good sky visibility and sections with tree coverage, but no buildings nearby. Uses a low-cost passive antenna. The median error decreases from 12.7 m to 5.2 m with tight FGO.

all experiments.

Interactive demos of the RTS smoother can be found at the bottom of the download views of all example tracks on the SnapperGPS website[5].

7.3.4 Discussion

If snapshots are captured at a relatively high rate of 1 Hz, then smoothing can roughly half the median localisation errors of the snapshot GNSS system in dynamic scenarios. Under good conditions, median errors down to 3 m are achieved. This is a good result considering that positioning is solely based on code-phase observations sampled at 4.092 MHz, which corresponds to a satellite-receiver distance resolution of $\frac{c}{4.092\,\mathrm{MHz}} \approx 73\,\mathrm{m}$. Overall, the median errors are higher than those reported in Chapters 3 and 4, but this can be explained by using lower-cost antennas for most of the recordings and the presence of more tree coverage.

All examined models and algorithms perform similar to each other in terms of localisation accuracy. A four-dimensional state-space model seems to slightly outperform a two-dimensional one due to its better ability to capture motion changes. However, the small gains might not justify the higher computational costs and the higher sensitivity w.r.t. parameter tuning. Furthermore, it is unclear to which extent the results for walking and cycling generalise to motion patterns of different animals. Furthermore, the results suggest that tightly coupled positioning and smoothing narrowly outperforms the loosely coupled approach. One reason for this is likely the better accuracy of fixes where there are some satellites captured, but not enough to resolve a position fix with standard snapshot GNSS. The loosely coupled approach discards the available satellite information and simply interpolates adjacent fixes. In contrast, the tightly coupled approach is able to make use of even a small number of observations, preserving information. In general, the differences between an RTS smoother and FGO are small because both methods solve the same optimisation problem—just using different optimisation techniques. Finally, GPR also performs well for high-rate data.

[5] `https://snappergps.info/view`

If the sampling interval is reduced to 10 s, then the benefits of smoothing are reduced. The median error shrinks by a about a third for the walking trials and only by about a quarter for the cycling trials when using smoothing in contrast to no smoothing. If the sampling interval is with 60 s even higher, then the benefit becomes marginal for walking and any advantage disappears for cycling. This is because of the movement dynamics being of higher frequency than the sampling frequency. The accuracy of Gaussian process regression in particular decreases for lower snapshot frequencies, but this might just be because of a sub-optimal kernel choice.

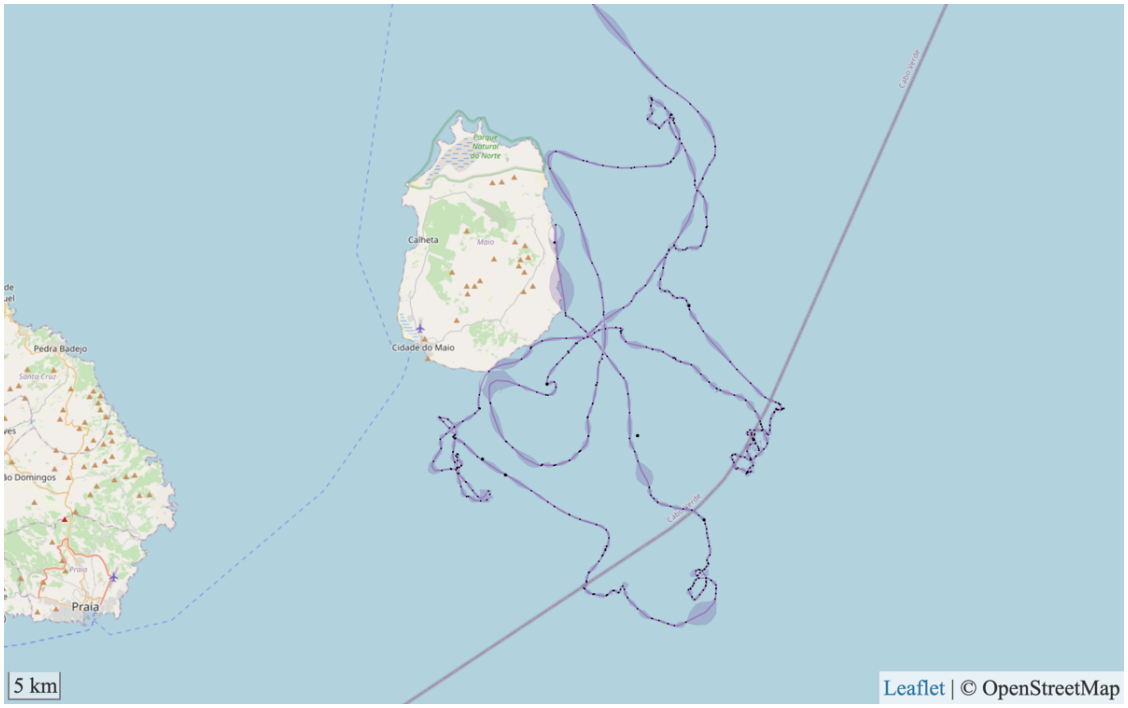


Figure 7.5: Gaussian process regression applied to a sea turtle track collected with a SnapperGPS receiver. Estimated means as pink line and 95% confidence intervals shaded in light pink. Non-smoothed SnapperGPS fixes as black circles.

Digression: Use of Smoothing for Low-Frequency Data in Practice
The results in Section 7.3.3 show that smoothing can only realise accuracy gains if data is collected at sufficiently short measurement intervals of less than a minute, at least for the applications scenarios examined, walking and cycling. However, smoothing can assist in certain scenarios where data is captured at lower rates, too. This box presents two such examples.

1. After the sea turtle deployment in 2021 (Section 5.2.1), cooperating biologists/ecologists were interested in the daily and total travel

distances of sea turtles tagged with SnapperGPS receivers. For this, only the low-frequency fixes when the turtle surfaced are available. The movement below the surface is entirely unknown and needs to be modelled. Assuming shortest-distance travel (a poly-line through all fixes) was deemed unrealistic by the experts while an interpolated curve (using GPR) was considered plausible, see Figure 7.5. The difference of the estimated travel distances over two weeks is 4 km: in total 466 km for the poly-line and 470 km for the Gaussian process curve. However, no ground truth is available to verify the accuracy of the estimate.

2. During a deployment on goats in 2022, the collaborator attached the SnapperGPS tags on the side of the individuals rather than at the top. This caused little sky visibility and unfavourable satellite geometries, with all visible satellites usually on one side of the sky. Consequently, the overall fraction of successful SnapperGPS fixes was low and the uncertainty of successful fixes high. However, since the speed of the goats was often slow, in contrast to the chosen measurement interval of 3 min, smoothing and interpolation can recover the tracks. Overlayed on satellite imagery, the estimated tracks are often within metres of trails and paths, suggesting that they are plausible. Tight FGO seems to be particularly beneficial since it allows one to still make use of satellite observations at points in time where less than five satellites are observed. However, ground truth from redundant GNSS receivers is again not available to quantify accuracy gains.

7.4 Conclusions

Several smoothing algorithms were designed and implemented to improve the accuracy of position estimates obtained with low-cost snapshot GNSS receivers. They were tested on several hours of walking and cycling data. All examined smoothing algorithms perform similarly well, with an approach that tightly couples snapshot positioning and smoothing performing best by a small margin. If data is collected in 1 Hz intervals, then localisation errors can be more than halved, down to median errors of about 3 m in dynamic tests under good conditions. This is despite using GNSS signal snapshots collected with the low-cost SnapperGPS system that only provides a pseudorange resolution of around 40 m or more. The accuracy is still comparable to commercial non-snapshot GNSS modules with higher energy consumption and more complex hardware, such as the i-GotU, which is popular for

low-cost wildlife tracking [7]. Furthermore, the decoupling of data acquisition and smoothing provides transparency about the underlying measurement and movement models and propagation of uncertainties and allows to adjust the model parameters in hindsight to address different movement behaviour. In contrast, conventional GNSS receivers often run a filtering algorithm on-board and only provide the filtered position fixes to the user (as a "black box").

However, the basic SnapperGPS system is designed primarily for longer wildlife tracking deployments lasting weeks or months, where its low energy consumption is advantageous, or for aquatic use, where the snapshot frequency is limited by the surfacing frequency of the animal. On the contrary, smoothing is shown to improve positioning accuracy only for shorter recordings lasting hours or days, where high-frequent snapshot capture is possible. Therefore, in practise, smoothing might only be beneficial in a few wildlife tracking scenarios:

- Short recordings with a high snapshot frequency, where snapshot GNSS is used despite it not being primarily designed for this use case.
- Tracking a very slowly moving animal, like a grazing goat, or one that frequently rests, like a tortoise.
- Refining fixes in an aquatic setting, where the snapshot rate is overall low, but snapshots appear bundled during brief surfacing events.

A fourth application scenario of the discussed algorithms is for fusion of snapshot GNSS data with data from other sensors. The HMMs and their factor-graph representations render it particularly easy to integrate additional observations through new measurement factors. In the case of wildlife tracking, additional observations could come from additional onboard sensors, such as an IMU or a barometer (cf. Chapter 3.2) and/or separate environmental monitoring stations, such as weather stations [149]. In addition, the models do not only apply to wildlife tracking and snapshot positioning, but could also be used for standard code-based positioning. Therefore, the next Chapter 8 adapts the best-performing

model and algorithm from this chapter for use in a sensor fusion application. No multi-sensor wildlife tracking platform with access to raw GNSS data is available at the time of writing. Therefore, Chapter 8 considers applications in robotics for the algorithm evaluation instead.

8

Factor Graph Fusion of Raw GNSS Sensing with IMU and Lidar for Precise Mobile Localisation without a Base Station

8.1 Problem Statement

This chapter considers more accurate localisation than the previous ones, which target low costs and low energy consumption rather than high accuracy. Accurate localisation is sometimes needed for wildlife tracking applications, e.g., when studying social interaction through proximity or when estimating an individual's energy expenditure or micro-movement [151–153]. In addition, it is ubiquitous in robotic applications [69, 77, 82, 83, 154, 155]. Therefore, this chapter covers localisation of mobile platforms in general. Furthermore, algorithms are considered that can run online in real time (often required in robotics) or offline (often sufficient in wildlife tracking).

As key enabler to improve accuracy, setups with multiple onboard sensors are considered. For these, effective sensor fusion is essential to maximise the information gain from each sensing modality. For example, proprioception based on inertial measurement units (IMUs) or encoders together with exteroception from cameras or lidars can be used to achieve a smooth, but slowly drifting, estimate of the motion

Figure 8.1: Setups for experimental validation. **Left:** a car driving on public roads. **Centre:** a quadruped robot moving in urban and forest environments. **Right:** a GNSS receiver carried handheld. Raw GNSS measurements are tightly fused with inertial sensing and optionally lidar using factor graphs.

of a mobile platform in a local environment. In contrast, satellite navigation can be used to estimate positions in a global Earth frame. These estimates are drift-free, but require sky visibility. Therefore, fusion of satellite navigation, proprioception, and exteroception is desirable for long-term operation, cf. Section 2.13.

The most popular way to fuse data from global navigation satellite systems (GNSS) with onboard perception sensing is two-stage fusion, cf. Section 2.13 and Section 7.2. First, an independent estimator running on the GNSS receiver computes global position fixes from raw GNSS observations. These fixes are then used as position priors within a second-stage pose estimator with proprioceptive and exteroceptive measurements [154, 155]. However, this two-stage fusion has several disadvantages, e.g., the two stages are only loosely coupled, which usually causes lower accuracy and higher uncertainty, especially, when a receiver acquires only a few satellites, or if the internal estimator dynamics of the first stage are unknown. Furthermore, maintaining a persistent network connection for differential GNSS (DGNSS) is impossible or inconvenient in some applications. Without DGNSS, receiver position uncertainty is in the order of several meters, cf. Section 2.11. Because of this, GNSS can only be used to anchor a platform's trajectory in the

global Earth frame or to correct long-term drift, but cannot aid accurate local navigation, e.g., if the platform's exteroception fails.

This chapter follows an alternative approach that addresses these disadvantages by instead fusing the raw observations of the individual satellites with proprioceptive and exteroceptive measurements in a single state estimator, which is more tightly coupled. This allows to leverage information from even just a few satellites (e.g., less than four)—information which would be discarded in the two-stage approach.

For each visible satellite and a given frequency band, a conventional GNSS receiver can provide three types of observables: pseudoranges, carrier phases, and Doppler shifts, cf. Section 2.8. These measurements are affected by a number of errors, rendering fusion particularly challenging and requiring robust optimisation. In addition, the different observables have distinct properties: pseudoranges have a comparably high uncertainty while carrier phases are accurate, but challenging to use in real time without a permanent communication link to a base station. The work in this chapter addresses both challenges. After a mathematical description of the system in Section 8.2, Section 8.3 introduces a novel factor graph design incorporating two types of raw GNSS observables (pseudroanges for drift-free localisation in the global Earth frame and differential carrier phases for accurate and smooth local positioning) together with inertial measurements and optionally lidar—to the best knowledge, the first factor graph design to do so. Section 8.4 covers the implementation of the algorithm using a single real-time optimisation phase, which implicitly handles GNSS initialisation, normal operation, and GNSS drop-out. This eliminates the need to switch between different modes for the aforementioned phases and leads to fast convergence. Finally, Section 8.5 presents an extensive evaluation on a car and a quadruped robot moving in challenging scenarios and comparison against the state-of-the-art on a public dataset.

8.2 System Description

The aim is to estimate the pose of a mobile platform that is equipped with a GNSS receiver, an IMU, and optionally a lidar—in real time and in a global

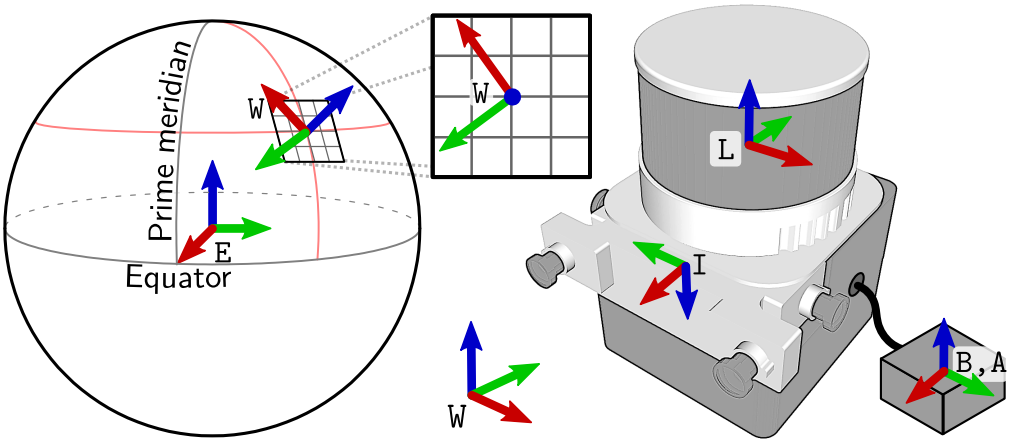


Figure 8.2: Reference frames: the Earth-centred-Earth-fixed (ECEF) frame E, local frame W, and frames on the platforms, including base B, which coincides with the GNSS antenna frame A, IMU frame I, and lidar frame L. (Figure courtesy of Marco Camurri.)

Earth frame. The estimated trajectory is required to be smooth/discontinuity-free. To mathematically describe this system, Section 8.2.1 introduces relevant coordinate frames, Section 8.2.2 defines the state of the system, Section 8.2.3 the measurements, and Section 8.2.4 the estimation problem.

8.2.1 Frames

Ultimately, position estimates in geographic coordinates (latitude, longitude) are of interest. However, for computational reasons, the system internally uses the Cartesian Earth-centred-Earth-fixed (ECEF) frame E as global frame. Its z-axis is the Earth's rotation axis, and its x-axis points to the prime meridian, cf. Figure 8.2.

If the system has exteroceptive sensors, then not only global position estimates are of interest, but also maintaining a smooth local trajectory with no discontinuities during initial GNSS convergence. Therefore, a local frame W is established that is aligned with the platform's pose at the start of the experiment. The optimisation estimates the transformation $\mathbf{T}_{\mathtt{EW}} \in \mathrm{SO}(3) \times \mathbb{R}^3$ between local frame W and global frame E jointly with the platform position in frame W. This ensures that the position estimate in W remains smooth even in scenarios where $\mathbf{T}_{\mathtt{EW}}$ converges slowly due to no or little GNSS observations being available at the start of an experiment.

If the sensing system has only GNSS and inertial sensing, then W is set to be identical to E. The frames rigidly attached to the robot are base B (coincident with the GNSS antenna frame A), IMU I, and lidar L, see Figure 8.2.

8.2.2 State

The state of the system at time t_i is

$$\boldsymbol{x}_i \triangleq [\mathbf{R}_i, \mathbf{p}_i, \mathbf{v}_i, \mathbf{b}_i^{\mathrm{g}} \ \mathbf{b}_i^{\mathrm{a}}, \delta\mathbf{t}_i^{\mathrm{r}}] \in \mathrm{SO}(3) \times \mathbb{R}^{12+M}, \tag{8.1}$$

where $\mathbf{R}_i = \mathbf{R}_{\mathtt{WB}}(t_i) \in \mathrm{SO}(3)$ and $\mathbf{p}_i = {}_{\mathtt{W}}\mathbf{p}_{\mathtt{WB}}(t_i) \in \mathbb{R}^3$ are the orientation and position of the base B with respect to W, respectively; $\mathbf{v}_i = {}_{\mathtt{B}}\mathbf{v}_{\mathtt{WB}}(t_i) \in \mathbb{R}^3$ is the linear velocity of B with respect to W expressed in B. The IMU's slowly changing gyroscope and accelerometer biases $\mathbf{b}_i^{\mathrm{g}}, \mathbf{b}_i^{\mathrm{a}} \in \mathbb{R}^3$ are expressed in frame I. Finally, $\delta\mathbf{t}_i^{\mathrm{r}} \in \mathbb{R}^M$ is the vector of the clock offsets between each satellite system and the receiver clock. The receiver accesses four satellite systems (i.e., $M = 4$): GPS, BeiDou, GLONASS, and Galileo.

If the setup includes a lidar, the optimisation estimates the transformation $\mathbf{T}_{\mathtt{EW}}$ in addition to the system states $\boldsymbol{x}_i$. The history of all unknowns is then

$$\mathcal{X}_k \triangleq \left\{ \{\boldsymbol{x}_i\}_{i \in \mathcal{T}_k}, \mathbf{T}_{\mathtt{EW}} \right\}, \tag{8.2}$$

where $\mathcal{T}_k$ is the set of all state indices in a fixed-length time window up to time t_k.

8.2.3 Measurements

The measurements include the proper acceleration and angular velocity in the IMU frame I and lidar point clouds $\mathcal{L}_{ij}$. Inertial measurements are received as a set between times t_i and t_j as $\mathcal{I}_{ij}$. They are preintegrated after gravity/bias compensation, as explained in Section 8.3.1. The GNSS receiver observes pseudoranges $\mathcal{P}_i$ and double-differenced carrier phases $\mathcal{C}_{ij}$. (The latter are explained in Section 8.3.4.) Thus, all measurements in a time window $\mathcal{T}_k$ are

$$\mathcal{Z}_k \triangleq \{\mathcal{I}_{ij}, \mathcal{L}_{ij}, \mathcal{P}_i, \mathcal{C}_{ij}\}_{i,j \in \mathcal{T}_k}. \tag{8.3}$$

The system creates a state $\boldsymbol{x}_i$ whenever there is a GNSS observation at time t_i and motion corrects a potentially received lidar point cloud at that timestamp [156]. If there is no GNSS observation for a certain amount of time, e.g., due to no sky visibility, the system creates a state with lidar and inertial measurements only.

8.2.4 Maximum-a-Posteriori Estimation

The optimisation maximises the likelihood of the measurements $\mathcal{Z}_k$, given the history of states $\mathcal{X}_k$

$$\mathcal{X}_k^* = \arg\max_{\mathcal{X}_k} p(\mathcal{X}_k|\mathcal{Z}_k) \propto p(\mathcal{X}_0)p(\mathcal{Z}_k|\mathcal{X}_k), \tag{8.4}$$

where measurements are assumed to be conditionally independent and corrupted by white Gaussian noise. Therefore, it can be expressed as a least-squares minimisation [69]

$$\begin{aligned}\mathcal{X}_k^* = \arg\min_{\mathcal{X}_k} & \|\mathbf{r}_0\|_{\Sigma_0}^2 \\ & + \sum_{i,j\in\mathcal{T}_k} \Bigg(\|\mathbf{r}_{\mathcal{I}_{ij}}\|_{\Sigma_{\mathcal{I}_{ij}}}^2 + \|\mathbf{r}_{\mathcal{L}_{ij}}\|_{\Sigma_{\mathcal{L}_{ij}}}^2 \\ & \qquad + \sum_{\rho_i^m\in\mathcal{P}_i} \|r_{\rho_i^m}\|_{\sigma_{\rho_i^m}}^2 + \sum_{\Delta\Delta\phi_{ij}^{mn}\in\mathcal{C}_{ij}} \|r_{\Delta\Delta\phi_{ij}^{mn}}\|_{\sigma_{\Delta\Delta\phi_{ij}^{mn}}}^2 \Bigg),\end{aligned} \tag{8.5}$$

where $\mathcal{T}_k$ is the set of all state indices in the sliding smoothing window up to time t_k. Each term is the residual associated with a factor type, weighted by the inverse of its covariance matrix. Residuals include a prior, IMU factors, relative odometry factors from lidar, and two types of GNSS factors, which are detailed in the following section.

8.3 Factor Graph Formulation

Figure 8.3 shows the factor graph structure. Each measurement factor is associated with a residual, which is the difference between a model-based prediction given the connected variables and the observation. Section 8.3.1 and 8.3.2 summarise the IMU and lidar residuals and before Section 8.3.3 and 8.3.4 detail the pseudorange and carrier-phase residuals.

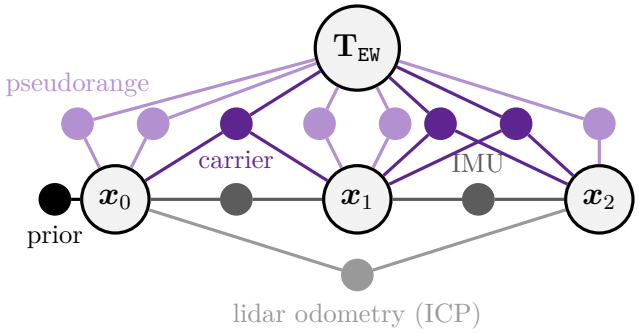


Figure 8.3: Factor graph structure with variable nodes (large circles) for states $\boldsymbol{x}_i$ and transformation $\mathbf{T}_{\mathtt{EW}}$ between local frame and global Earth frame and factor nodes (small, coloured) for priors, and properioceptive (IMU), exteroceptive (lidar), and GNSS measurements (pseudorange, carrier phase).

8.3.1 Pre-Integrated Inertial Measurements

The standard method of IMU measurement pre-integration is used to constrain the pose, velocity, and biases between two consecutive nodes x_i and $x_{i+1} = x_j$, providing high-frequency state updates between nodes. For a description of the residual $\mathbf{r}_{\mathcal{I}_{ij}} \in \mathbb{R}^{15}$, see Forster et al. [157].

8.3.2 ICP Registration

The lidar measurements are registered to a local submap [156] using iterative closest point (ICP) odometry [158] and the registration output is added as relative pose factor between times t_i and t_j with the residual

$$\mathbf{r}_{\mathcal{L}_{ij}} = \Phi\left(\tilde{\mathbf{T}}_i^{-1}\tilde{\mathbf{T}}_j^{-1}\mathbf{T}_i^{-1}\mathbf{T}_j\right), \tag{8.6}$$

where $\mathbf{T}_i = [\mathbf{R}_i, \mathbf{p}_i]$ is the estimated pose, $\tilde{\mathbf{T}}_i \in \mathrm{SE}(3)$ is the estimate from the ICP module, and Φ is the lifting operator defined by Forster et al. [157].

8.3.3 Pseudoranges

For each acquired satellite signal m at time t_i, a GNSS receiver reports[1] a pseudorange $\rho_i^m \in \mathcal{P}_i$, which is the observed signal travel time multiplied by the speed of light, cf. Section 2.8. It is close to the spatial satellite-receiver distance, but affected

[1] Or, in case of a snapshot receiver, a pseudorange is reconstructed from the code phase found in the raw GNSS signal snapshot provided by the receiver, cf. Section 2.5.

by additional terms, in particular signal delays in different layers of the atmosphere and time offsets of the clocks that are used to measure the signal travel time [59]. The pseudorange residual for a single received satellite signal m is approximately

$$\begin{aligned} r_{\rho_i^m} = &\left\| \mathbf{s}_i^m - \mathbf{T}_{\mathtt{EW}} \mathbf{p}_i \right\| \\ &+ \delta\rho_{\mathrm{T}} \left(\mathbf{s}_i^m, \mathbf{T}_{\mathtt{EW}} \mathbf{p}_i \right) + \delta\rho_{\mathrm{I}} \left(\mathbf{s}_i^m, \mathbf{T}_{\mathtt{EW}} \mathbf{p}_i \right) \\ &+ c \cdot \delta t_{i,g}^{\mathrm{r}} - \left(\rho_i^m + c \cdot \left(\delta t^m + \nu^m \right) \right), \end{aligned} \tag{8.7}$$

where $\mathbf{s}_i^m \in \mathbb{R}^3$ is the satellite position in frame $\mathtt{E}$ at the signal transmit time that corresponds to the observation time t_i. It is corrected for the Earth rotation. The spatial signal delay in the troposphere is $\delta\rho_{\mathrm{T}} \colon \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}_0^+$, the delay in the ionosphere is $\delta\rho_{\mathrm{I}} \colon \mathbb{R}^3 \times \mathbb{R}^3 \to \mathbb{R}_0^+$, the speed of light is c, and the offset of the receiver clock from the clock of the GNSS with index $g \in \{0 \dots 3\}$ is $\delta t_{i,g}^{\mathrm{r}} \in \mathbb{R}$. The pseudorange ρ_i^m is adjusted to correct for satellite clock bias $\delta t^m \in \mathbb{R}$ and relativity $\nu^m \in \mathbb{R}$.

Satellite position, satellite clock offset, and both atmospheric delays are modelled based on data broadcasted by the satellites. Due to these effects, modelling errors can be up to a few meters. The antenna position and the receiver clock offset are the unknowns that need to estimated. If the system uses a multi-band receiver, then there can be multiple residuals for a single satellite, one for each band in which the receiver acquired a signal. Alternatively, observations from different bands can be combined to estimate atmospheric delays.

8.3.4 Carrier Phases

The residual of carrier phase $\phi_i^m \in \mathbb{R}^+$ (in units of lengths) could be written similarly to the pseudorange residual [59]

$$\begin{aligned} r_{\phi_i^m} = &\left\| \mathbf{s}_i^m - \mathbf{T}_{\mathtt{EW}} \mathbf{p}_i \right\| \\ &+ \delta\rho_{\mathrm{T}} \left(\mathbf{s}_i^m, \mathbf{T}_{\mathtt{EW}} \mathbf{p}_i \right) - \delta\rho_{\mathrm{I}} \left(\mathbf{s}_i^m, \mathbf{T}_{\mathtt{EW}} \mathbf{p}_i \right) + \lambda^m \omega^m \\ &+ c \cdot \delta t_{i,g}^{\mathrm{r}} - \left(\phi_i^m + \lambda^m N_i^m + c \cdot \left(\delta t^m + \nu^m \right) \right), \end{aligned} \tag{8.8}$$

see Section 2.8. Most of the terms are the same as in Equation (8.7). However, the delay in the ionosphere $\delta\rho_{\mathrm{I}}$ has the opposite sign. There is also a wind-up effect

$\omega^m \in \mathbb{R}$ resulting from the interplay between the changing satellite orientation and the circularly polarised carrier wave (with wavelength $\lambda^m \in \mathbb{R}^+$), which has a magnitude ranging from centimetres to a few decimetres [61]. The most significant difference is an unknown offset in the number $N_i^m \in \mathbb{N}$ of wavelengths.

The advantage of carrier-phase observations over pseudorange ones is that their measurement noise is only $\sim$5 mm. However, modelling them precisely in real time is challenging because satellite positions and atmospheric delays are not necessarily known with submetre accuracy. Furthermore, the integer ambiguity N_i^m is an additional unknown that needs to be estimated. Therefore, double-differencing is applied, i.e., multiple observations are combined to cancel out terms that are unknown or imprecisely known. Specifically, the algorithm combines two observations of two different satellites of the same GNSS and signal band at the current time t_i with two older observations of the same satellites at a time $t_i < t_j$ [83]

$$\Delta\Delta\phi_{ij}^{mn} = \left(\tilde{\phi}_j^n - \tilde{\phi}_j^m\right) - \left(\tilde{\phi}_i^n - \tilde{\phi}_i^m\right), \tag{8.9}$$

where $\tilde{\phi}_i^m = \phi_i^m + c \cdot (\delta t^m + \nu^m)$ is the carrier phase ϕ_i^m corrected for satellite clock bias and relativity. If the change of the atmospheric delays of the satellite signals m and n between times t_i and t_j is negligible and the signals are still locked (i.e., $N_i^m = N_j^m$ and $N_i^n = N_j^n$), then the residual is

$$\begin{aligned} r_{\Delta\Delta\phi_{ij}^{mn}} = & \left(\|\mathbf{s}_j^n - \mathbf{T}_{\mathtt{EW}}\mathbf{p}_j\| - \|\mathbf{s}_j^m - \mathbf{T}_{\mathtt{EW}}\mathbf{p}_j\|\right) \\ & - \left(\|\mathbf{s}_i^n - \mathbf{T}_{\mathtt{EW}}\mathbf{p}_i\| - \|\mathbf{s}_i^m - \mathbf{T}_{\mathtt{EW}}\mathbf{p}_i\|\right) \\ & - \Delta\Delta\phi_{ij}^{mn}. \end{aligned} \tag{8.10}$$

For each pair of band and satellite system (GPS, GLONASS, Galileo, BeiDou), the satellite that has been continuously visible for the longest time is selected for the first satellite signal m. For the second satellite signal n, the algorithm iterates over all remaining satellite observations in the same signal band. It creates a residual for each satellite pair obtained this way.

Table 8.1: Mean (top) and median (bottom) horizontal localisation errors [m] in the global Earth frame with RTK as ground truth.

			baseline methods				proposed method	
dataset	**platform**	**duration [s]**	**GNSS-fix**	**IMU, GNSS-fix**	**IMU, ICP, GNSS-fix**	**GVINS**	**IMU, raw-GNSS**	**IMU, ICP, raw-GNSS**
HK	car	2454				1.75	**1.54**	
						1.63	**1.44**	
Jericho	car	923	4.01	4.04	failure		2.22	**2.03**
			2.67	2.82			1.88	**1.71**
Park Town	car	264	4.57	4.46			**2.79**	
			2.56	2.54			**2.00**	
Bagley	quadruped	1120	3.40	3.46	4.78		2.34	**2.07**
			2.84	3.16	5.50		**1.97**	2.02
Thom	quadruped	440	12.31	10.93	4.24		1.66	**1.33**
			7.42	9.32	2.92		**0.86**	1.00

8.4 Implementation

The GNSS factors from Section 8.3.3 and 8.3.4 are integrated into the VILENS state estimator, which includes IMU and lidar registration factors already and runs in real time [156]. Lidar registration factors are created at half the rate of the GNSS factors and the factor graph is solved with a fixed lag smoother based on the incremental optimiser iSAM2 [71] in the GTSAM library [69]. The transformation $\mathbf{T}_{\mathtt{EW}}$ between the local and global frame is continuously estimated, while the states outside the optimisation window $\mathcal{T}_k$ are marginalised.

Raw GNSS observations are prone to outliers; for example, the multi-path effect occurs when satellite signals are reflected by surrounding buildings or vegetation before they are received. This induces a longer travel distance than the direct line of sight. To mitigate this, a threshold-based outlier detector is applied first and then the Huber loss function is used to reduce the effect of remaining outliers [77].

The GNSS processing components are implemented using the GPSTk library [159]. To avoid a cold start of at least 30 s where satellite positions and clocks are unknown, publicly available satellite navigation data is preloaded before the start of an experiment. No online data is used thereafter.

8.5 Evaluation

To evaluate the proposed algorithm, three experiments are conducted using both, public and self-recorded datasets.

Experiment 1 (Section 8.5.1 - raw GNSS and IMU fusion): Comparison against the state-of-the-art using the *urban driving* sequence of the public GVINS dataset [82].

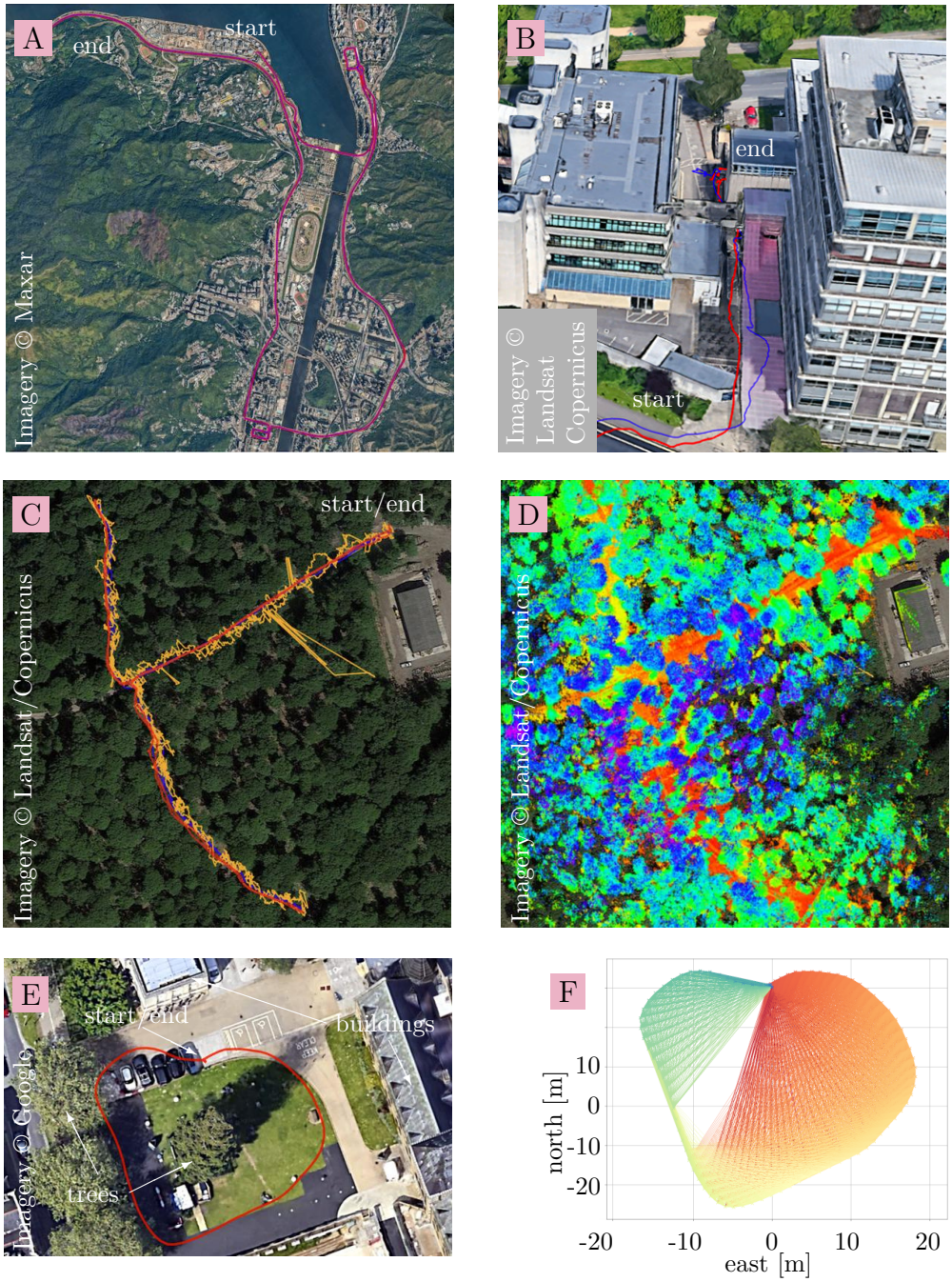


Figure 8.4: Experimental results. **(A)**: Trajectory of a car in Hong Kong estimated with the algorithm proposed in this chapter fusing inertial and GNSS measurements (red) compared to RTK (ground truth, blue) [82]. **(B)**: Trajectory of a quadruped traversing between two buildings estimated using IMU, ICP, and GNSS (red) compared to RTK (blue), part of sequence *Thom*. The RTK trajectory drifted into the building when there was little GNSS data while estimates of the proposed algorithm did not. **(C)**: Trajectory of a quadruped in the Bagley Wood estimated using IMU, ICP, and GNSS (red) in comparison to RTK (blue) and single GNSS fixes (orange). **(D)**: Same trajectory with lidar scans overlayed. **(E)**: Trajectory of a handheld device estimated using time-differential carrier-phase factors only and given the starting position. **(F)**: The corresponding factors between different states in time, represented by lines and crosses respectively. Few factors were created to states near the trees on the left. Still, the horizontal localisation error at the end was less than 10 cm.

The setup for this dataset included a 200 Hz consumer-grade Analog Devices ADIS16448 IMU, two 20 Hz Aptina MT9V034 cameras, and a 10 Hz entry-level multi-band u-blox C099-F9P GNSS receiver. Because there was no lidar data, only the fusion of IMU and raw GNSS was evaluated with the algorithm proposed in this chapter. The sequence has a duration of 41 min and includes sections with little sky visibility (urban canyons) and brief complete GNSS drop-outs (underpasses). In addition, four sequences with limited sky visibility were collected in and around Oxford, UK by mounting a 5 Hz C099-F9P GNSS receiver with a multi-band GNSS antenna and a consumer-grade 200 Hz Bosch BMI085 IMU on an electric vehicle and a Boston Dynamics Spot quadruped robot, see Figure 8.1. The sequences evaluated were:

- *Jericho*: a 4.4 km driving loop through narrow streets in Oxford's city centre at speeds up to 11 m/s. This includes several sections with GNSS drop-out.
- *Park Town*: a 1.2 km drive along tree-lined avenues.
- *Bagley*: the quadruped moving through a dense commercial forest. This is known to be challenging for satellite navigation due to the very limited sky visibility, many outliers in the GNSS measurements because of signal reflections by surrounding vegetation (multi-path effect), and signal degradation caused by the electromagnetic interference of the robot with the GNSS signals. This is quantified by two measures: first, the pseudorange observations of the receiver have on average four-times higher standard deviations and, second, on average 25% less satellites are visible, both in comparison to *Jericho*.
- *Thom*: the quadruped walked around a high-rise building, passed through a tunnel for 45 s and ended in a yard between tall buildings, see Figure 8.4-B.

Experiment 2 (Section 8.5.2 - raw GNSS, IMU, and lidar fusion): This experiment evaluated fusion with data from a 10 Hz Hesai XT32 lidar that was part of the setup for the sequences *Jericho*, *Bagley*, and *Thom*.

Experiment 3 (Section 8.5.3 - carrier-phase-only fusion): Finally, this experiment tested the carrier-phase factor's usefulness for accurate local navigation. For this, the GNSS receiver was hand carried around a 12 m tall tree for 105 m and 2 min, cf. Figure 8.4-E to create the sequence *Tree*.

For all sequences, RTK was used to obtain ground truth.

8.5.1 Experiment 1: Raw GNSS and IMU fusion

The column **GVINS** of Table 8.1 shows the localisation errors of the open-source GVINS algorithm, which fuses pseudoranges and Doppler shifts from the GNSS receiver with inertial measurements and vision constraints from a camera. The column **IMU, raw-GNSS** presents results for the proposed algorithm fusing pseudoranges, carrier phases, and inertial measurements only. The proposed algorithm performs slightly better than GVINS despite the fact that it does not use a camera. This demonstrates that carrier-phase observations can replace exteroception for accurate local navigation when the latter is unavailable, but some sky visibility exists.

There is a practical difference between the algorithms, too: GVINS requires a dedicated initialisation phase with sufficient sky visibility that lasts for 15.3 s for the *urban driving* sequence. In contrast, the proposed algorithm does not require such a separate phase and converges to the correct pose of `B` in the Earth frame `E` as soon as the first slight motion occurs, after less than 4 s. This is likely partially due to the utilisation of carrier phases: a small motion is sufficient to estimate the orientation of `B` in `E` from the precise differential carrier phases, while more motion is required to achieve the same with the less accurate pseudoranges and Doppler shifts.

Table 8.1 compares the accuracy of separately computed non-differential GNSS fixes (column **GNSS-fix**), fusion of these fixes with inertial measurements (**IMU, GNSS-fix**), and the proposed algorithm fusing raw GNSS observations with inertial measurements (**IMU, raw-GNSS**) for the newly recorded sequences. The median global accuracy is 1–2 m, which is sufficient for many autonomous vehicle applications, e.g., to initialise a fine-grained lidar localisation system or to reject

incorrect place proposals from such a system. The accuracy is also close to that achieved on the *Hong Kong* sequence despite the GNSS receiver operating at half the rate and the newly recorded sequences being shorter and having no sections with as good sky visibility. Both issues hinder global convergence. Furthermore, the trajectory estimates are smooth after initial convergence, as can be seen in the supplementary material[1]. In particular, the 30–60 % smaller error on *Bagley*[1] demonstrates the advantage of using individual GNSS observations and inertial measurements in a single optimisation framework as opposed to separating the computation of GNSS fixes and the fusion. The gains mainly come from improved robustness in this scenario, which has fewer and more noisy GNSS observations.

To summarise, these results show that the pseudorange factors enable robust localisation in the global Earth frame. In addition, the carrier-phase factors help to create a locally smooth and accurate trajectory when exteroception is absent.

8.5.2 Experiment 2: Raw GNSS, IMU, and Lidar Fusion

For the sequences *Jericho*, *Bagley*[1], and *Thom*, the accuracy of fusing separately computed GNSS fixes with IMU measurements and ICP (**IMU, ICP, GNSS-fix**) is also compared to the proposed algorithm fusing raw GNSS observations with inertial measurements and ICP (**IMU, ICP, raw-GNSS**) in Table 8.1. The global accuracy is similar to *Experiment 1* because lidar measurements only provide local information. However, a georeferenced map of the local environment is also obtained, see Figure 8.4-D. Furthermore, the results for *Thom* show that the proposed algorithm, with its single optimisation stage, can implicitly handle GNSS drop-out and smoothly switch between GNSS-aided and non-GNSS navigation, cf. Figure 8.4-B. The baseline methods fusing GNSS fixes drift more in this case, cf. Table 8.1.

8.5.3 Experiment 3: Carrier-Phase-Only Fusion

Lastly, the proposed algorithm[2] was tested with only double-differential carrier-phase factors and without proprioception or exterioception on the *Tree* sequence, see Figure 8.4-E and 8.4-F. Here the double-differenced carrier-phase measurements provided an input roughly equivalent to relative local odometry. The horizontal positioning error in the end was less than 10 cm, indicating performance close to Suzuki's method [83], despite that the proposed algorithm only uses one type of GNSS factor and had no clear sky visibility, unlike Suzuki in his experiments [83].

8.6 Conclusions

This chapter presented a novel system to localise a mobile platform in an unknown environment in real time by tightly fusing GNSS observations, proprioceptive measurements, and optionally exteroceptive measurements into a single factor graph optimisation. The approach employs raw GNSS observations from individual satellites instead of pre-computed position fixes to maximise the information taken from the GNSS receiver. This includes not only pseudorange observations for absolute positioning on Earth, but also accurate carrier-phase observations for precise local localisation to support the situation where exteroceptive measurements are either unavailable or degraded. This chapter showed that the proposed approach improves accuracy by up to several meters in comparison to the two-stage algorithms where a pre-computed fix is used in the factor graph instead of raw data—especially if the view of the sky is limited. Typical median localisation errors in the Earth frame are still just 1–2 m.

The open-source implementation of a part of the algorithm earned almost 100 stars on the source-code hosting platform GitHub, indicating the relevance of the proposed system for the community.

[2]Open-source code to reproduce the results in Figure 8.4-E and 8.4-F is provided on `https://github.com/JonasBchrt/raw-gnss-fusion` together with the dataset sequences *Bagley* and *Tree* and interactive maps with the estimated trajectories corresponding to some of the results in Table 8.1.

9 Conclusions

9.1 Summary

The core of this thesis presented the design of a low-cost, low-power, open-source snapshot GNSS system, which is primarily intended for animal tracking in the wild. The first step towards the realisation of the system was covered in Chapter 3: the development of tailored probabilistic algorithms that enable reliable location estimation from short GNSS signal snapshots with ultra-low resolution. These include estimation and correction of hardware error parameters as well as careful satellite selection for robust and sufficiently accurate localisation. The algorithms run fast enough to process large amounts of real-world data in a scalable and economical way. They are open-source and implemented in a publicly accessible, easy-to-use web application.

Chapter 4 presented the second core component of the system: snapshot GNSS receiver hardware that is low-cost, low-power, robust, open-source, and easy to manufacture. The availability of the aforementioned robust algorithms was a core enabler for the hardware, allowing the use of low-power and low-cost components, including an RF front-end for around $1. This resulted in a complete receiver that can cost less than $30 and can log data for a location fix every 25 min for more than a year while operating on two 0.6 g button cells or a 1.2 g LiPo battery.

Besides its low hardware requirements, a fundamental difference between the developed system and pure snapshot GNSS systems described in other publications is the additional effort that was invested into making it accessible to a large number of end-users with various deployment scenarios and demonstrating robust localisation in these scenarios. Hardware and software are fully open-source and extensively documented, which allowed several individuals to independently replicate it. In addition, a three-digit number of receivers was distributed to end-users—mostly conservationists and researchers—and deployed on various species ranging from sea turtles to venomous snakes, see Chapter 5. The web application and documentation allowed users to independently configure and deploy their receivers and to process and visualise their data. User feedback was constantly considered to iteratively improve the system and build customised versions, including daughterboards for collection of accelerometer data (Chapter 4), triggering signal acquisition for surfacing aquatic animals (Chapter 4), and wireless near real-time offloading of data (Chapter 6). The feedback and the deployments reviewed in Chapter 5 provide evidence that the developed snapshot GNSS system benefits longer deployments on small animals that cannot carry bulky batteries, studies with larger sample size on a constraint budget, tracking of aquatic animals that surface only briefly, and customisation of tags to cater specific deployment needs. It also seems to have the potential to support uptake of wildlife tracking by conservationists and researchers for whom costs were previously too high, especially, in the Global South.

Chapter 7 presented a tightly coupled smoothing method using incremental factor graph optimisation to improve the localisation accuracy of low-cost, low-power snapshot GNSS down to a few metres for deployments where data is captured at higher rates. This extends the application area of snapshot GNSS from long wildlife tracking deployments, where low power consumption is more important than high accuracy, to short deployments with more demanding accuracy requirements. Unlike conventional GNSS receivers, the snapshot GNSS approach decouples data acquisition from smoothing by moving all computations from the device to the cloud. It provides transparency about the smoothing's underlying measurement

and movement models and propagation of uncertainties and allows to adjust model parameters in the hindsight to address different movement behaviour. This is unlike conventional GNSS receivers, which often operate as a "black box" by running a filtering algorithm on-board and only providing filtered position fixes to the user.

Chapter 8 also addressed the "black-box" behaviour of conventional GNSS receivers. It showed that the smoothing algorithm can be modified to build a novel system to localise a general mobile platform in an unknown environment in real time by tightly fusing raw GNSS observations, proprioceptive measurements, and optionally exteroceptive measurements in a single factor graph optimisation. Unlike the aforementioned snapshot GNSS system, this includes not only pseudorange observations for absolute positioning on Earth, but also accurate carrier-phase observations for precise local localisation. The proposed approach improves accuracy by up to several meters in comparison to conventional two-stage algorithms where a pre-computed location fix is fused with the other sensing modalities instead of raw GNSS data—especially if the view of the sky is limited.

Both systems were not only useful for the projects in which context they were initially developed, but also as open-source tools for GNSS research projects conducted by third parties. In addition, both provide evidence that separating data acquisition and compute can have advantages over the established concept of fully-integrated GNSS receivers that perform signal acquisition, positioning, outlier detection, and filtering in a single module. In certain scenarios, the separation approach can enable either more energy-efficient or more robust localisation.

9.2 Outlook

The user feedback covered by Chapter 5 arrived with suggestions for future work on the snapshot GNSS system that could potentially be impactful and increase the number of use cases for which it could be employed. This includes further reduction of the weight of the hardware. An important step into this direction would be to replace the antennas of the receivers with light-weight PCB antennas. However, the attempts in Chapter 4 revealed that the signal quality that is provided by

this class of antennas is too low if no further hardware changes are implemented. Therefore, it may be worth it to develop a version with more expensive components (e.g., an MCU with more RAM and/or functionality to acquire signals at a higher sampling rate) to increase sampling frequency and/or snapshot length to improve signal quality and enable the use of smaller antennas.

Further future work could introduce and evaluate additional features such as snapshot capturing triggered by an activity sensor or the combination with additional sensors.

Crucially, everyone can participate in the further development of the system due to the open-source nature of the project and its documentation.

On the software side, the integration of additional types of GNSS signals, like GPS L1C and GLONASS L1OCM, promises improved robustness and accuracy in the future, once they are widely broadcasted. The system's cloud-based approach allows to implement these changes without any hardware modifications or firmware updates. Therefore, all current users can benefit from future algorithm updates—unlike users of conventional GNSS receivers that estimate locations on-board.

In addition to animal tracking, third parties use the open-source software and algorithms developed alongside this thesis for research and development in at least two other application areas, and future work could focus on tailored systems for those. The first area is on-board localisation on hand-held or wearable devices. This setting does not depend on cloud processing. Instead, ephemerides are obtained via an existing 4G or 5G cellular down-link or an auxiliary traditional GNSS chip set, and all processing is performed locally. For this application, it would probably also be advisable to use slightly more complex hardware to improve signal quality compared to the signal quality provided by the low-cost RF front-end of the receivers presented in Chapter 4. This could reduce computation time and/or antenna size, potentially rendering the technique feasible to be implemented on a hand-held or wrist-worn device. The core motivation behind using snapshot GNSS in this application domain is to save energy and, thus, increase battery life and/or fix frequency, by significantly reducing signal capturing times in contrast to traditional GNSS or A-GNSS.

Furthermore, the snapshot technique cannot only be used with GNSS signals, but also with signals from satellites that are not intended for positioning. For example, conventional GNSS techniques do not work with the modern STARLINK satellite constellation and its thousands of satellites. In contrast, snapshot GNSS has the potential to enable STARLINK-based positioning, despite the system being designed for internet access, not positioning [160]. However, this would require a hardware platform that captures signals in the appropriate signal band. In general, applying snapshot GNSS to non-GNSS satellite signals seems to be a promising research direction, which also requires algorithms that cannot only work with coarse time, but also with lower signal quality and less precise measurements when acquiring signals not designed for positioning. An application area are satellite receivers designed for reliable positioning even with interference in some or all of the GNSS signal bands, e.g., through jamming or spoofing.

References

[1] Laura A. McMahon et al. “Evaluation of micro-GPS receivers for tracking small-bodied mammals”. In: *PLOS ONE* 12.3 (Mar. 2017). Ed. by Mathew S. Crowther. URL: https://doi.org/10.1371/journal.pone.0173185.

[2] Patrick W. Cain and Matthew D. Cross. “An open-source hardware GPS data logger for wildlife radio-telemetry studies: a case study using Eastern box turtles”. In: *HardwareX* 3 (2018). https://doi.org/10.1016/j.ohx.2018.02.002, pp. 82–90.

[3] Devan Allen McGranahan, Benjamin Geaumont, and Jonathan W. Spiess. “Assessment of a livestock GPS collar based on an open-source datalogger informs best practices for logging intensity”. In: *Ecology and Evolution* 8.11 (2018). https://doi.org/10.1002/ece3.4094, pp. 5649–5660.

[4] Alison Matthews et al. “The success of GPS collar deployments on mammals in Australia”. In: *Australian Mammalogy* 35.1 (2013), pp. 65–83. URL: https://doi.org/10.1071/AM12021.

[5] Blake M Allan et al. “A cost-effective and informative method of GPS tracking wildlife”. In: *Wildlife Research* 40.5 (2013), pp. 345–348. URL: https://doi.org/10.1071/WR13069.

[6] Kate Moses et al. “An introduction to satellite technologies for tracking wildlife”. In: (Feb. 2023). URL: https://www.wildlabs.net/article/download-now-best-practice-guide-satellite-technologies-tracking-wildlife.

[7] Gail Morris and L Mike Conner. “Assessment of accuracy, fix success rate, and use of estimated horizontal position error (EHPE) to filter inaccurate data collected by a common commercially available GPS logger”. In: (2017). URL: https://doi.org/10.1371/journal.pone.0189020.

[8] Andrew Markham. “On a wildlife tracking and telemetry system: a wireless network approach”. PhD thesis. University of Cape Town, South Africa, 2008. URL: http://hdl.handle.net/11427/5172.

[9] Saskia Wischnewski et al. “Variation in foraging strategies over a large spatial scale reduces parent–offspring conflict in Manx shearwaters”. In: *Animal Behaviour* 151 (2019), pp. 165–176. URL: https://doi.org/10.1016/j.anbehav.2019.03.014.

[10] E. J. Critchley et al. “Assessing the effectiveness of foraging radius models for seabird distributions using biotelemetry and survey data”. In: *Ecography* 43.2 (2020), pp. 184–196. URL: https://doi.org/10.1111/ecog.04653.

[11] Ben Dean et al. "Behavioural mapping of a pelagic seabird: combining multiple sensors and a hidden Markov model reveals the distribution of at-sea behaviour". In: *Journal of The Royal Society Interface* 10.78 (2013). URL: https://doi.org/10.1098/rsif.2012.0570.

[12] Timm A Wild et al. "A multi-species evaluation of digital wildlife monitoring using the Sigfox IoT network". In: *Animal Biotelemetry* 11.1 (2023), pp. 1–17. URL: https://doi.org/10.1186/s40317-023-00326-1.

[13] Lorenzo Quaglietta et al. "A low-cost GPS GSM/GPRS telemetry system: performance in stationary field tests and preliminary data on wild otters (Lutra lutra)". In: *PLoS ONE* 7.1 (Jan. 2012). Ed. by Rohan H. Clarke. https://doi.org/10.1371/journal.pone.0029235, pp. 1–10.

[14] Timothy C. A. Molteno. "Estimating position from millisecond samples of GPS signals (the "FastFix" algorithm)". In: *Sensors* 6 (2020). URL: https://doi.org/10.3390/s20226480.

[15] Timm A Wild et al. "Micro-sized open-source and low-cost GPS loggers below 1 g minimise the impact on animals while collecting thousands of fixes". In: *Plos one* 17.6 (2022). URL: https://doi.org/10.1371/journal.pone.0267730.

[16] Hilary R. Kauth et al. "Low-cost DIY GPS trackers improve upland game bird monitoring". In: *Wildlife Biology* 2020.2 (2020). https://doi.org/10.2981/wlb.00653.

[17] Mark Hebblewhite and Daniel T. Haydon. "Distinguishing technology from biology: a critical review of the use of GPS telemetry data in ecology". In: *Philosophical Transactions of the Royal Society B: Biological Sciences* 365.1550 (2010), pp. 2303–2312. URL: https://doi.org/10.1098/rstb.2010.0087.

[18] Graeme C Hays et al. "Key questions in marine megafauna movement ecology". In: *Trends in ecology & evolution* 31.6 (2016), pp. 463–475. URL: https://doi.org/10.1016/j.tree.2016.02.015.

[19] Frank van Diggelen. *A-GPS: assisted GPS, GNSS, and SBAS*. Norwood, MA, USA: Artech House, 2009. URL: https://ieeexplore.ieee.org/document/9100796.

[20] Timothy C. A. Molteno and Keith W. Payne. "FastFix albatross data: snapshots of raw GPS L-1 data from Southern Royal Albatross". In: *Data* 6.4 (2021). URL: https://doi.org/10.3390/data6040037.

[21] Jonas Beuchert and Alex Rogers. "SnapperGPS: algorithms for energy-efficient low-cost location estimation using GNSS signal snapshots". In: *Proceedings of the 19th ACM Conference on Embedded Networked Sensor Systems*. SenSys '21. Coimbra, Portugal: Association for Computing Machinery, 2021, pp. 165–177. URL: https://doi.org/10.1145/3485730.3485931.

[22] Amanda Matthes et al. "SnapperGPS: deployment of a low-cost snapshot GNSS receiver to track loggerhead sea turtles". In: *Proceedings of the 40th International Sea Turtle Symposium (ISTS40)*. International Sea Turtle Society. 2022. URL: https://ora.ox.ac.uk/objects/uuid:c9acf083-d5e5-4265-8425-67509c5e3b9b.

[23] Jonas Beuchert, Amanda Matthes, and Alex Rogers. "SnapperGPS: open hardware for energy-efficient, low-cost wildlife location tracking with snapshot GNSS". In: *Journal of Open Hardware* 7.1 (2023). URL: https://doi.org/10.5334/joh.48.

[24] Jonas Beuchert, Marco Camurri, and Maurice Fallon. "Factor graph fusion of raw GNSS sensing with IMU and lidar for precise robot localization without a base station". In: *IEEE International Conference on Robotics and Automation (ICRA)*. 2023, pp. 8415–8421. URL: https://doi.org/10.1109/ICRA48891.2023.10161522.

[25] Herval Silva et al. *The conservation of migratory marine vertebrates: new insights into habitat use, threats, and behaviour of sea turtles*. Submitted to the International Scientific Symposium on the Canary Current Large Marine Ecosystem (under review). 2023.

[26] James B.-Y. Tsui. *Fundamentals of Global Positioning System receivers*. New York, NY, USA: John Wiley & Sons, Inc., Nov. 2004. URL: https://doi.org/10.1002/0471712582.

[27] Kai Borre et al. *A software-defined GPS and Galileo receiver: a single-frequency approach*. New York, NY, USA: Springer Science & Business Media, 2007. URL: https://doi.org/10.1007/978-0-8176-4540-3.

[28] Philip Kwan. *NAVSTAR GPS space segment/navigation user segment interfaces (IS-GPS-200K)*. National Coordination Office for Space-Based Positioning, Navigation, and Timing. Mar. 2019. URL: https://www.gps.gov/technical/icwg/IS-GPS-200K.pdf (visited on 05/06/2020).

[29] China Satellite Navigation Office. "BeiDou navigation satellite system signal in space interface control document open service signal B1C (version 1.0)". In: (Dec. 2017). URL: http://en.beidou.gov.cn/SYSTEMS/Officialdocument/201806/P020180608525871869457.pdf.

[30] Shulin Yan, Shenghong Zhou, and Yuguo Zhang. "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)*. Istanbul, Turkey: IEEE, 2018, pp. 46–53. URL: https://doi.org/10.1109/CATA.2018.8398654.

[31] Kelly C. Seals. "Enhanced acquisition techniques for GPS L1C receivers". PhD thesis. Worcester Polytechnic Institute, 2014. URL: https://digital.wpi.edu/concern/etds/4f16c3039.

[32] Jie Liu et al. "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, 2012, pp. 85–98. URL: https://doi.org/10.1145/2426656.2426666.

[33] Per K. Enge. "The Global Positioning System: signals, measurements, and performance". In: *International Journal of Wireless Information Networks* 1.2 (1994), pp. 83–105.

[34] C. Fernández–Prades et al. “GNSS-SDR: an open source tool for researchers and developers”. In: *Proceedings of the 24th International Technical Meeting of The Satellite Division of the Institute of Navigation*. Portland, Oregon, Sept. 2011, pp. 780–794. URL: https://www.ion.org/publications/abstract.cfm?articleID=9640.

[35] David Akopian. “Fast FFT based GPS satellite acquisition methods”. In: *IEE Proceedings–Radar, Sonar and Navigation* 152.4 (2005), pp. 277–286. URL: https://doi.org/10.1049/ip-rsn:20045096.

[36] Bashir A. Siddiqui et al. “Joint data-pilot acquisition and tracking of Galileo E1 open service signal”. In: *Ubiquitous Positioning Indoor Navigation and Location Based Service*. Finland: IEEE, 2010, pp. 1–7. URL: https://doi.org/10.1109/UPINLBS.2010.5654005.

[37] Andrew Simsky and Frank Boon. “Carrier phase and Doppler-based algorithms for real-time standalone positioning”. In: *Proceedings of GNSS*. Graz, Austria: Austrian Institute of Navigation, 2003, pp. 1–25.

[38] Clyde C. Goad and L. Goodman. “A modified tropospheric refraction correction model”. In: *American Geophysical Union Annual Fall Meeting*. San Francisco, CA, USA, Dec. 1974.

[39] John A Klobuchar. “Ionospheric effects on GPS”. In: *Global Positioning System: theory and application* 136 (1996), pp. 513–514.

[40] *Minimum operational performance standards for global positioning system/wide area augmentation system airborne equipment*. RTCA Inc. 2006.

[41] Jaume Sanz Subirana, José M. Juan Zornoza, and Mamuel Hernández-Pajares. *Best linear unbiased minimum-variance estimator (BLUE)*. European Space Agency. 2011. URL: https://gssc.esa.int/navipedia/index.php/Best_Linear_Unbiased_Minimum-Variance_Estimator_(BLUE) (visited on 04/30/2020).

[42] *Snapshot positioning - next generation GNSS receiver for low power applications*. Baseband Technologies Inc. URL: http://www.basebandtech.com/wp-content/uploads/Snapshot-Receiver.pdf (visited on 03/24/2020).

[43] Ignacio Fernández-Hernández and Kai Borre. “Snapshot positioning without initial information”. In: *GPS Solutions* 20.4 (Mar. 2016), pp. 605–616. URL: https://doi.org/10.1007/s10291-016-0530-4.

[44] Pau Closas Gómez. “Bayesian signal processing techniques for GNSS receivers: from multipath mitigation to positioning”. PhD thesis. Universitat Politècnica de Catalunya, 2009. URL: https://upcommons.upc.edu/bitstream/handle/2117/94259/PCG.pdf.

[45] Penina Axelrad et al. “Collective detection and direct positioning using multiple GNSS satellites”. In: *Navigation* 58.4 (2011), pp. 305–321. URL: https://doi.org/10.1002/j.2161-4296.2011.tb02588.x.

[46] Pascal Bissig, Manuel Eichelberger, and Roger Wattenhofer. “Fast and robust GPS fix using one millisecond of data”. In: *16th ACM/IEEE International Conference on Information Processing in Sensor Networks (IPSN)*. New York, NY, USA: ACM Press, 2017, pp. 223–234. URL: https://doi.org/10.1145/3055031.3055083.

[47] European GNSS Agency (GSA). *Power-efficient positioning for the Internet of Things: merging GNSS with low-power connectivity solutions: white paper*. 2020. URL: https://doi.org/10.2878/437669.

[48] Benjamin Peterson, Richard Hartnett, and Geffrey Ottman. "GPS receiver structures for the urban canyon". In: *Proceedings of the 8th International Technical Meeting of the Satellite Division of the Institute of Navigation (ION GPS 1995)*. 1995, pp. 1323–1332. URL: https://www.ion.org/publications/abstract.cfm?articleID=2454.

[49] Emanuele Ziglioli and Eugenio Realini. "Extending goGPS for snapshot positioning". In: *Geomatics Workbooks* 12 (2015), pp. 383–393.

[50] Manuel Eichelberger, Ferdinand von Hagen, and Roger Wattenhofer. "Multi-year GPS tracking using a coin cell". In: *Proceedings of the 20th International Workshop on Mobile Computing Systems and Applications*. HotMobile '19. Santa Cruz, CA, USA: ACM, Feb. 2019, pp. 141–146. URL: https://doi.org/10.1145/3301293.3302367.

[51] Ambuj Varshney. "Making low-power and long-range wireless backscatter transmitters". In: *GetMobile: Mobile Computation and Communication* 26.2 (July 2022), pp. 5–11. URL: https://doi.org/10.1145/3551670.3551672.

[52] Minkang Wang, Honglei Qin, and Tian Jin. "Massive terminal positioning system with snapshot positioning technique". In: *GPS Solutions* 23 (2019), pp. 1–14. URL: https://doi.org/10.1007/s10291-018-0821-z.

[53] i-gotU. *GT-120B vs GT-600B*. 2022. URL: https://cdn.shopify.com/s/files/1/2104/5147/files/GT-120BvsGT-600B.pdf.

[54] Advanced Telemetry Systems Inc. *W510 Wildlink GPS Logger*. 2023. URL: https://atstrack.com/tracking-products/transmitters/W510-wildlink-gps-logger.aspx.

[55] Lotek Wireless Inc. *MicroGPS+*. 2023. URL: https://www.lotek.com/products/microgps/.

[56] Telonics Inc. *GPS/store on board terrestrial systems*. 2019. URL: https://www.telonics.com/products/gps4/gps-sob.php.

[57] Conrad J. Foley and Claudio Sillero-Zubiri. "Open-source, low-cost modular GPS collars for monitoring and tracking wildlife". In: *Methods in Ecology and Evolution* 11.4 (2020). https://doi.org/10.1111/2041-210X.13369, pp. 553–558.

[58] Craig Morrison. *ATS G10 Ultralite GPS*. Advanced Telemetry Systems Australia. URL: https://nebula.wsimg.com/2c21477b3a3e90c8279c4b5979312195?AccessKeyId=CEA7BFCD16F82D670903&disposition=0 (visited on 03/24/2020).

[59] Elliott Kaplan and Christopher Hegarty. *Understanding GPS: principles and applications*. Artech House, 2005.

[60] Bradford W. Parkinson et al. *Global Positioning System: theory and applications, Volume II*. Washington, DC, USA: American Institute of Aeronautics and Astronautics, 1996. URL: https://doi.org/10.2514/4.866395.

[61] Jiun-Tsong Wu et al. "Effects of antenna orientation on GPS carrier phase". In: *Astrodynamics* (1992), pp. 1647–1660.

[62] u-blox AG. *AssistNow*. June 2021. URL: https://www.u-blox.com/en/technologies/agnss-assistnow.

[63] Baseband Technologies Inc. *28 day extended ephemeris*. URL: https://www.basebandtech.com/28-day-extended-ephemeris/.

[64] JF Zumberge et al. "Precise point positioning for the efficient and robust analysis of GPS data from large networks". In: *Journal of geophysical research: solid earth* 102.B3 (1997), pp. 5005–5017. URL: https://doi.org/10.1029/96JB03860.

[65] Jan Kouba, François Lahaye, and Pierre Tétreault. "Precise point positioning". In: *Springer Handbook of Global Navigation Satellite Systems*. Springer, 2017, pp. 723–751. URL: https://doi.org/10.1007/978-3-319-42928-1_25.

[66] Peter J. G. Teunissen, Paul J. De Jonge, and Christian C. J. M. Tiberius. "The LAMBDA method for fast GPS surveying". In: *International Symposium on GPS Technology Applications*. 1995.

[67] Anette Rietdorf, Christopher Daub, and Peter Loef. "Precise positioning in real-time using navigation satellites and telecommunication". In: *Proceedings of The 3rd Workshop on Positioning and Communication*. Citeseer. 2006.

[68] Bernd Eissfeller et al. "Real-time kinematic in the light of GPS modernization and Galileo". In: *Proceedings of the 14th International Technical Meeting of the Satellite Division of The Institute of Navigation*. 2001, pp. 650–682. URL: https://www.ion.org/publications/abstract.cfm?articleID=1749.

[69] Frank Dellaert, Michael Kaess, et al. "Factor graphs for robot perception". In: *Foundations and Trends in Robotics* 6.1-2 (2017), pp. 1–139. URL: https://dx.doi.org/10.1561/2300000043.

[70] Michael Kaess, Ananth Ranganathan, and Frank Dellaert. "iSAM: incremental smoothing and mapping". In: *IEEE Transactions on Robotics* 24.6 (2008), pp. 1365–1378. URL: https://doi.org/10.1109/TRO.2008.2006706.

[71] Michael Kaess et al. "iSAM2: incremental smoothing and mapping using the Bayes tree". In: *Intl. Journal of Robotics Research* 31.2 (2012), pp. 216–235. URL: https://doi.org/10.1177/0278364911430419.

[72] Weisong Wen et al. "It is time for factor graph optimization for GNSS/INS integration: comparison between FGO and EKF". In: *arXiv: 2004.10572* (2020). URL: https://doi.org/10.48550/arXiv.2004.10572.

[73] Daniel Wilbers, Christian Merfels, and Cyrill Stachniss. "A comparison of particle filter and graph-based optimization for localization with landmarks in automated vehicles". In: *2019 Third IEEE International Conference on Robotic Computing (IRC)*. 2019, pp. 220–225. URL: https://doi.org/10.1109/IRC.2019.00040.

[74] Jay A. Farrell. *Aided navigation: GPS with high rate sensors*. McGraw-Hill, 2008.

[75] Paul David Groves. *Principles of GNSS, inertial, and multisensor integrated navigation systems*. Artech House, 2013.

[76] Ryan M. Watson and Jason N. Gross. "Evaluation of kinematic precise point positioning convergence with an incremental graph optimizer". In: *IEEE/ION Position, Location and Navigation Symposium (PLANS)*. 2018, pp. 589–596. URL: https://doi.org/10.1109/PLANS.2018.8373431.

[77] Ryan M. Watson. "Enabling robust state estimation through covariance adaptation". PhD thesis. Morgantown: West Virginia Univ., 2019. URL: https://doi.org/10.33915/etd.7455.

[78] Weisong Wen et al. "Tightly coupled GNSS/INS integration via factor graph and aided by fish-eye camera". In: *IEEE Transactions on Vehicular Technology* 68.11 (2019), pp. 10651–10662. URL: https://doi.org/10.1109/TVT.2019.2944680.

[79] Niko Sünderhauf and Peter Protzel. "Towards robust graphical models for GNSS-based localization in urban environments". In: *IEEE International Multi-Conference on Systems, Signals & Devices*. 2012, pp. 1–6. URL: https://doi.org/10.1109/SSD.2012.6198059.

[80] Niko Sünderhauf et al. "Switchable constraints and incremental smoothing for online mitigation of non-line-of-sight and multipath effects". In: *IEEE Intelligent Vehicles Symposium*. 2013, pp. 262–268. URL: https://doi.org/10.1109/IVS.2013.6629480.

[81] Zheng Gong et al. "Tightly coupled integration of GNSS and vision SLAM using 10-DoF optimization on manifold". In: *IEEE Sensors Journal* 19.24 (2019), pp. 12105–12117. URL: https://doi.org/10.1109/JSEN.2019.2935387.

[82] Shaozu Cao, Xiuyuan Lu, and Shaojie Shen. "GVINS: tightly coupled GNSS–visual–inertial fusion for smooth and consistent state estimation". In: *IEEE Trans. on Robotics* (2022), pp. 1–18. URL: https://doi.org/10.1109/TRO.2021.3133730.

[83] Taro Suzuki. "Time-relative RTK-GNSS: GNSS loop closure in pose graph optimization". In: *IEEE Robotics and Automation Letters* 5.3 (2020), pp. 4735–4742. URL: https://10.1109/LRA.2020.3003861.

[84] Tomoji Takasu and Akio Yasuda. "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB". In: *International Symposium on GPS/GNSS*. 2009. URL: https://gpspp.sakura.ne.jp/paper2005/isgps_2009_rklib.pdf.

[85] Woosik Lee et al. "Tightly-coupled GNSS-aided visual-inertial localization". In: *International Conference on Robotics and Automation (ICRA)*. 2022. URL: https://doi.org/10.1109/ICRA46639.2022.9811362.

[86] H. D. J. N. Aldridge and R. M. Brigham. "Load Carrying and Maneuverability in an Insectivorous Bat: a Test of the 5% "Rule" of Radio-Telemetry". In: *Journal of Mammalogy* 69.2 (May 1988), pp. 379–382. URL: https://doi.org/10.2307/1381393.

[87] GPSWebShop. *i-gotU GT-120B GPS / GNSS data logger*. 2022. URL: https://gpswebshop.com/products/i-gotu-gt-120b-travel-sports-gps-data-logger.

[88] Joachim Claudet and Simonetta Fraschetti. "Human-driven impacts on marine habitats: a regional meta-analysis in the Mediterranean Sea". In: *Biological Conservation* 143.9 (2010), pp. 2195–2206. URL: https://doi.org/10.1016/j.biocon.2010.06.004.

[89] Stefanie Werner et al. "Harm caused by marine litter". In: (2016). URL: https://data.europa.eu/doi/10.2788/690366.

[90] Graeme C. Hays and Lucy A. Hawkes. "Satellite tracking sea turtles: opportunities and challenges to address key questions". In: *Frontiers in Marine Science* 5 (2018). URL: https://doi.org/10.3389/fmars.2018.00432.

[91] David A. Gill et al. "Capacity shortfalls hinder the performance of Marine Protected Areas globally". In: *Nature* 543.7647 (2017), pp. 665–669. URL: https://doi.org/10.1038/nature21708.

[92] Jane Lubchenco and Kirsten Grorud-Colvert. "Making waves: the science and politics of ocean protection". In: *Science* 350.6259 (2015), pp. 382–383. URL: https://doi.org/10.1126/science.aad5443.

[93] PJ Bouchet and JJ Meeuwig. "Drifting baited stereo-videography: a novel sampling tool for surveying pelagic wildlife in offshore marine reserves". In: *Ecosphere* 6.8 (2015), pp. 1–29. URL: https://doi.org/10.1890/ES14-00380.1.

[94] Freya C Womersley et al. "Global collision-risk hotspots of marine traffic and the world's largest fish, the whale shark". In: *Proceedings of the National Academy of Sciences* 119.20 (2022), e2117440119. URL: https://doi.org/10.1073/pnas.2117440119.

[95] Gail Schofield et al. "Evidence-based marine protected area planning for a highly mobile endangered marine vertebrate". In: *Biological Conservation* 161 (2013), pp. 101–109. URL: https://doi.org/10.1016/j.biocon.2013.03.004.

[96] José J Lahoz-Monfort and Michael JL Magrath. "A comprehensive overview of technologies for species and habitat monitoring and conservation". In: *BioScience* 71.10 (2021), pp. 1038–1062. URL: https://doi.org/10.1093/biosci/biab073.

[97] Michael R Donaldson et al. "Making connections in aquatic ecosystems with acoustic telemetry monitoring". In: *Frontiers in Ecology and the Environment* 12.10 (2014), pp. 565–573. URL: https://doi.org/10.1890/130283.

[98] Antoine M. Dujon, R. Todd Lindstrom, and Graeme C. Hays. "The accuracy of Fastloc-GPS locations and implications for animal tracking". In: *Methods in Ecology and Evolution* 5.11 (2014), pp. 1162–1169. URL: https://doi.org/10.1111/2041-210X.12286.

[99] Jan-Olaf Meynecke and Nikolai Liebsch. "Asset Tracking Whales—First Deployment of a Custom-Made GPS/GSM Suction Cup Tag on Migrating Humpback Whales". In: *Journal of Marine Science and Engineering* 9.6 (2021). URL: https://doi.org/10.3390/jmse9060597.

[100] Katherine L Mansfield et al. "Satellite tag attachment methods for tracking neonate sea turtles". In: *Marine Ecology Progress Series* 457 (2012), pp. 181–192. URL: https://doi.org/10.3354/meps09485.

[101] Radiocommunication Sector of the International Telecommunication Union. "Electrical characteristics of the surface of the Earth". In: (June 2017). URL: https://www.itu.int/dms_pubrec/itu-r/rec/p/R-REC-P.527-4-201706-I!!PDF-E.pdf.

[102] Keith Van Dierendonck, Ming Xu, and Pejman L. Kazemi. *System, method, and computer program for a low power and low cost GNSS receiver*. US Patent 9,116,234. Aug. 2015.

[103] Martin A. Fischler and Robert C. Bolles. "Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography". In: *Communications of the ACM* 24.6 (1981), pp. 381–395. URL: https://doi.org/10.1145/358669.358692.

[104] Ryan M. Watson et al. "Enabling robust state estimation through measurement error covariance adaptation". In: *IEEE Transactions on Aerospace and Electronic Systems* 56.3 (2019), pp. 2026–2040. URL: https://doi.org/10.1109/TAES.2019.2941103.

[105] James J. Spilker Jr. *Digital communications by satellite*. Englewood Cliffs, NJ, USA: Prentice Hall, 1977.

[106] DMA WGS 84 Development Committee. *Department of Defense World Geodetic System 1984: its definition and relationships with local geodetic systems*. Tech. rep. Fairfax, VA, USA: United States. Defense Mapping Agency, 1987. URL: https://apps.dtic.mil/sti/citations/ADA280358.

[107] Frank G. Lemoine et al. *The development of the joint NASA GSFC and NIMA geopotential model EGM96*. Tech. rep. Greenbelt, MD, USA: NASA Goddard Space Flight Center, 1998.

[108] Nikolaos K. Pavlis et al. "The development and evaluation of the Earth Gravitational Model 2008 (EGM2008)". In: *Journal of geophysical research: solid earth* 117.B4 (2012). URL: https://doi.org/10.1029/2011JB008916.

[109] Christopher Amante and Barry W. Eakins. *ETOPO1 arc-minute global relief model: procedures, data sources and analysis*. Vol. NESDIS NGDC-24. Boulder, CO, USA: National Geophysical Data Center, 2009. URL: https://doi.org/10.7289/V5C8276M.

[110] Earth Resources Observation And Science (EROS) Center. *Shuttle radar topography mission (SRTM) 1 arc-second global*. 2017. URL: 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.

[111] Duracell Batteries. *76A / LR44 alkaline manganese dioxide battery*. Visited on 28 May 2023. URL: https://www.duracell.com/wp-content/uploads/2018/06/76ALR44V1_062018.pdf.

[112] Donald G Archer and Peiming Wang. "The dielectric constant of water and Debye-Hückel limiting law slopes". In: *Journal of Physical and Chemical Reference Data* 19.2 (1990), pp. 371–411. URL: https://doi.org/10.1063/1.555853.

[113] Steven W Ellingson. *Electromagnetics, Volume 1*. VT Publishing, 2018. URL: https://doi.org/10.21061/electromagnetics-vol-1.

[114] *Magnetic materials*. URL: https://www.microwaves101.com/encyclopedias/magnetic-materials.

[115] Silcon Labs. "Hardware design for capacitive touch". In: *AN0040 - Application Note* (2013). URL: https://www.silabs.com/documents/public/application-notes/AN0040.pdf.

[116] Silicon Laboratories Inc. *EFM32HG reference manual*. 2015. URL: %5Curl%7Bhttps://www.silabs.com/documents/public/reference-manuals/efm32hg-rm.pdf%7D%20(accessed%2016%20March%202020).

[117] *RS Pro LR44 alkaline coin battery, 1.5 V, 158 mAh*. URL: https://docs.rs-online.com/e11c/0900766b81650940.pdf.

[118] Isidor Buchmann. *Batteries in a portable world: a handbook on rechargeable batteries for non-engineers*. eng. 4. edition. Richmond, British Columbia: Cadex Electronics Inc, 2016.

[119] International Union for Conservation of Nature and Natural Resources. *The IUCN red list of threatened species*. Feb. 2022. URL: https://www.iucnredlist.org/.

[120] Juan Patino-Martinez et al. "Strategic nest site selection in one of the world's largest loggerhead turtle nesting colonies, on Maio Island, Cabo Verde". In: *Oryx* 57.2 (2023), pp. 152–159. URL: https://doi.org/10.1017/S0030605321001496.

[121] Amanda Matthes. "Tracking Manx Shearwaters with SnapperGPS". In: *EPSRC Centre for Doctoral Training in Autonomous Intelligent Machines & Systems - ANNUAL REVIEW 2021/22* (2022), pp. 17–18. URL: https://aims.robots.ox.ac.uk/media/12113/aims-annual-review-202122-web.pdf.

[122] *Human ageing genomic resources*. URL: https://genomics.senescence.info/.

[123] *Encyclopedia of life*. URL: https://eol.org/.

[124] The World Bank Group. *World Bank country and lending groups*. URL: https://datahelpdesk.worldbank.org/knowledgebase/articles/906519-world-bank-country-and-lending-groups.

[125] Dyna Rochmyaningsih. "Indonesian plan to clamp down on foreign scientists draws protest". In: *Nature News* (May 2018). URL: https://www.nature.com/articles/d41586-018-05001-7.

[126] Dyna Rochmyaningsih. "Indonesia's strict new biopiracy rules could stifle international research". In: *Science* (July 2019). URL: https://www.science.org/content/article/indonesia-s-strict-new-biopiracy-rules-could-stifle-international-research.

[127] Roland Kays et al. "Terrestrial animal tracking as an eye on life and planet". In: *Science* 348.6240 (2015). URL: https://doi.org/10.1126/science.aaa2478.

[128] Alison Matthews et al. "The success of GPS collar deployments on mammals in Australia". In: *Australian Mammalogy* 35 (2013), pp. 65–83. URL: https://doi.org/10.1071/AM12021.

[129] Jake Wall et al. "Novel opportunities for wildlife conservation and research with real-time monitoring". In: *Ecological Applications* 24.4 (2014), pp. 593–601. URL: https://doi.org/10.1890/13-1971.1.

[130] GS Karthick, M Sridhar, and PB Pankajavalli. "Internet of Things in animal healthcare (IoTAH): review of recent advancements in architecture, sensing technologies and real-time monitoring". In: *SN Computer Science* 1 (2020), pp. 1–16. URL: https://doi.org/10.1007/s42979-020-00310-z.

[131] M Gor et al. "GATA: GPS-Arduino based tracking and alarm system for protection of wildlife animals". In: *2017 International Conference on Computer, Information and Telecommunication Systems (CITS)*. IEEE. 2017, pp. 166–170. URL: https://doi.org/10.1109/CITS.2017.8035325.

[132] Tile Inc. *Tile Pro (2022)*. 2023. URL: https://www.tile.com/product/black-pro.

[133] Bernd Heidtmann. *Low-power GNSS for tracking applications*. u-blox AG. URL: https://content.u-blox.com/sites/default/files/Low-power-tracking-applications_WhitePaper_UBX-21014570%5C%281%5C%29.pdf.

[134] u-blox AG. *Ultra-low-power asset tracking*. June 2022. URL: https://www.u-blox.com/en/ultra-low-power-asset-tracking.

[135] Ashish Kumar Sultania et al. "Energy modeling and evaluation of NB-IoT with PSM and eDRX". In: *2018 IEEE Globecom Workshops*. IEEE. 2018, pp. 1–7. URL: https://doi.org/10.1109/GLOCOMW.2018.8644074.

[136] Svetoslav Duhovnikov et al. "Power consumption analysis of NB-IoT technology for low-power aircraft applications". In: *2019 IEEE 5th World Forum on Internet of Things (WF-IoT)*. IEEE. 2019, pp. 719–723. URL: https://doi.org/10.1109/WF-IoT.2019.8767234.

[137] Stefano Parrino, Giacomo Peruzzi, and Alessandro Pozzebon. "LoPATraN: low power asset tracking by means of narrow band IoT (NB-IoT) technology". In: *Sensors* 21.11 (2021), p. 3772. URL: https://doi.org/10.3390/s21113772.

[138] The Things Network. *LoRaWAN airtime calculator*. URL: https://www.thethingsnetwork.org/airtime-calculator.

[139] Ferran Adelantado et al. "Understanding the limits of LoRaWAN". In: *IEEE Communications Magazine* 55.9 (2017), pp. 34–40. URL: http://ieeexplore.ieee.org/document/8030482/ (visited on 02/06/2023).

[140] Petroc Taylor. "Average cellular data price per gigabyte in the United States 2018-2023". In: (Jan. 2023). URL: https://www.statista.com/statistics/994913/average-cellular-data-price-per-gigabyte-in-the-us/.

[141] Global mobile Suppliers Association (GSA). *2G-3G shutdowns: a comprehensive guide*. 2021. URL: https://gsacom.com/paper/2g-3g-shutdowns-a-comprehensive-guide/.

[142] Ltd. Quectel Wireless Solutions Co. *Quectel BG95 series LTE Cat M1/ Cat NB2/ EGPRS module*. 2020. URL: https://www.quectel.com/wp-content/uploads/pdfupload/Quectel_BG95_Series_LPWA_Specification_V1.5.pdf.

[143] Nordic Semiconductor. *What is cellular IoT?* URL: https://www.nordicsemi.com/Products/Low-power-cellular-IoT/What-is-cellular-IoT.

[144] GSM Association. *Mobile & massive IoT*. en-US. URL: https://www.gsma.com/iot/massive-iot/ (visited on 05/07/2023).

[145] GSM Association. *Mobile IoT deployment map*. en-US. May 2023. URL: https://www.gsma.com/iot/deployment-map/ (visited on 05/07/2023).

[146] Ray Ozzie et al. *Blues Wireless Notecard datasheet SKU: NOTE-NBGL-500*. Blues Inc., Jan. 2022. URL: https://dev.blues.io/hardware/notecard-datasheet/note-nbgl-500/.

[147] Ray Ozzie et al. *Blues Wireless Notecarrier datasheet*. Blues Inc., Apr. 2022. URL: https://dev.blues.io/hardware/notecarrier-datasheet/notecarrier-a/.

[148] Kenshi Saho. "Kalman filter for moving object tracking: performance analysis and filter design". In: *Kalman Filters*. Ed. by Ginalber Luiz de Oliveira Serra. Rijeka: IntechOpen, 2017. Chap. 12. URL: https://doi.org/10.5772/intechopen.71731.

[149] Raphaël Nussbaumer et al. "Reconstructing bird trajectories from pressure and wind data using a highly optimized hidden Markov model". In: *Methods in Ecology and Evolution* 14.4 (2023), pp. 1118–1129. URL: https://doi.org/10.1111/2041-210X.14082.

[150] Christopher KI Williams and Carl Edward Rasmussen. *Gaussian processes for machine learning*. Vol. 2. 3. MIT Press Cambridge, MA, 2006. URL: https://doi.org/10.7551/mitpress/3206.001.0001.

[151] Gabriella Gall. "Group coordination and decision-making during foraging in meerkats (Suricata suricatta)". PhD thesis. University of Zurich, June 2017. URL: https://doi.org/10.5167/uzh-138436.

[152] Bart Kranstauber. "Modelling animal movement as Brownian bridges with covariates". In: *Movement ecology* 7.1 (2019), pp. 1–10. URL: https://doi.org/10.1186/s40462-019-0167-3.

[153] Pritish Chakravarty. "Sensor and the beast: generalised methods to recognise animal behaviour and quantify energy expenditure using inertial sensors, and applications". PhD thesis. EPFL, 2020. URL: https://doi.org/10.5075/epfl-thesis-7579.

[154] Tixiao Shan et al. "LIO-SAM: tightly-coupled lidar inertial odometry via smoothing and mapping". In: *2020 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. Las Vegas, NV, USA: IEEE Press, 2020, pp. 5135–5142. URL: https://doi.org/10.1109/IROS45743.2020.9341176.

[155] Zheng Gong et al. "Graph-based adaptive fusion of GNSS and VIO under intermittent GNSS-degraded environment". In: *IEEE Transactions on Instrumentation and Measurement* 70 (2021), pp. 1–16. URL: https://doi.org/10.1109/TIM.2020.3039640.

[156] David Wisth, Marco Camurri, and Maurice Fallon. "VILENS: visual, inertial, lidar, and leg odometry for all-terrain legged robots". In: *IEEE Trans. on Robotics* (Aug. 2022), pp. 1–18. URL: https://doi.org/10.1109/TRO.2022.3193788.

[157] Christian Forster et al. "On-manifold preintegration for real-time visual-inertial odometry". In: *IEEE Trans. on Robotics* 33.1 (2017), pp. 1–21. URL: https://doi.org/10.1109/TRO.2016.2597321.

[158] François Pomerleau et al. "Comparing ICP variants on real-world data sets". In: *Autonomous Robots* 34.3 (2013), pp. 133–148. URL: https://doi.org/10.1007/s10514-013-9327-2.

[159] R Benjamin Harris and Richard G Mach. "The GPSTk: an open source GPS toolkit". In: *GPS Solutions* 11.2 (2007), pp. 145–150. URL: https://doi.org/10.1007/s10291-006-0043-7.

[160] Todd E Humphreys et al. “Signal structure of the Starlink Ku-band downlink”. In: *IEEE Transactions on Aerospace and Electronic Systems* (2023). URL: https://doi.org/10.1109/TAES.2023.3268610.