SDR Technology Updates

Kai Borre

The GNSS Laboratory at

Samara State Aerospace University
Global Navigation Satellite Systems (GNSS)

- GPS
- GLONASS
- BeiDou
- Galileo

All systems transmit on two or more frequencies
Why Use Software Defined Receivers?

- They offer a flexible open-architecture, are easily upgradeable, and are reconfigurable in real-time where dynamic changes of parameters is possible (loop noise bandwidth, damping ration of DLL, correlator spacing, acquisition thresholds like bandwidth, sampling IF, and elevation mask).

There is a variety of terminology in use. In this presentation we use the term **GNSS software receiver about complete receivers, mostly written in Matlab, but also a few in C.**
Software Defined Receivers (SDR)

- Matlab based, non-real time

- Simulink based, real time when tracking four PRNs

- FPGA and GPP (C code), real time
SDR Front-ends

- SiGe, L1 frequency
- Maxim, L1 frequency
- sdrnav40, four eligible frequencies
L1 and code only

- Matlab GPS
- Matlab Galileo
- Matlab GLONASS
- Matlab BeiDou
- Matlab SBAS channel
SDR Plots

Real correlation result from GNSS SDR

3rd Sogei Workshop, January 21–22  © 2016 by Kai Borre
More SDR Plots

- Xilinx GPS L1 code and phase
- Xilinx GPS L1 & L5 code and phase
Snap-shot Receivers
Snap-shot Positioning

The technique is based on saving digital samples from an ADC of a GNSS receiver front-end. The samples are postprocessed on a server which computes a position and time solution. The samples typically are 10–20 ms long and are thus power saving. The postprocessing on a remote allows various security controls.

Obvious applications include tracking of (stolen) items, rescue activities in remote areas, road tolling, etc.

We present an algorithm (designed by Ignacio Fernández-Hernández) which allows computing position and time without any initial position or time information. We are motivated by computing positions with minimum amount of prior information.
Snap-shot Receiver

Compact, low power snapshot device (rover)

RF front-end
- Amplifier
- Mixer
- A/D
- Frequency synthesizer
- Receiver clock

Signal recording
- Memory

GNSS antenna

Data Aiding
- DGPS
- SBAS
- Multi-frequency, multisystem GNSS receiver
- ...

Software that does GNSS signal processing, derives measurements and does the actual position computation

Additional tasks can be precision improvement or GNSS signal validation

PVT solution
A GNSS signal just acquired cannot establish synchronization between system and receiver times. A synchronization is only possible after demodulation of the signal and identification of a given bit pattern (TLM).

A complete decoding of the ephemeris and computation of the satellite position takes 30 s. A first position is obtained after 30–60 s.

The snap-shot technique can provide an instantaneous position. The receiving device is cheaper than an ordinary GNSS receiver and requires less power. The proposed algorithm yields accuracies of a few meters and no prior knowledge of position.
Code Phase Ambiguity Resolution

Classical GNSS positioning exploits the pseudorange $P$ observation

$\text{(received time} - \text{transmission time)} \times c.$

A snap-shot receiver does not know the number of full 1 ms codes. That is, we need to estimate a multiple of 300 km contained in the pseudorange.
Pseudorange presented as true range plus receiver clock offset, and as an integer of GPS 1-ms codes plus the code phase observation.
Coarse Time Positioning

Let $e^i$ be the unit vector in direction of satellite-receiver, $dt$ the receiver clock offset, $t_c$ difference between estimated and actual observation time for satellite $i$ (coarse time difference), and $\dot{\rho}^i = -(D^i - f_d)\lambda$ is the range rate between satellite and receiver. The doppler value is $D^i$, and the clock frequency drift is denoted $f_d$. The linearized observation equations for pseudoranges are [unit: m]

\[
b = \begin{bmatrix} p^i_{\text{obs}} & \cdots & p^i_{\text{comp}} \\ \vdots & \ddots & \vdots \\ \vdots & \ddots & \vdots \end{bmatrix} = Ax + e = \begin{bmatrix} \cdots & 1 & \dot{\rho}^i \\ -e^i & \cdots \\ \vdots & \cdots & c \, dt \\ t_c \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ t_c \end{bmatrix} + e. \tag{1}
\]
The receiver cannot know for sure the correct integer of 1-ms codes until it verifies the residuals of the position solution.

Equation (1), by Peterson, works in case the initial time deviates ±1 minute from the true time and in a vicinity of ±150 km of the true position.
Taylor Expansion of Range Rate

\[ \dot{\rho}^i(t_1) \approx \dot{\rho}^i(t_0) + (t_1 - t_0)\ddot{\rho}^i(t_0). \] (2)

The quantity \( \dot{\rho}^i(t_0) \) is illustrated below and \( \ddot{\rho}^i(t_0) \) on the next slide:

Range rates \( \dot{\rho}^i \) of visible GPS satellites on March 2, 2010 at Danish GPS Center, Aalborg, Denmark.
Satellite-receiver accelerations $\ddot{p}^i$ of visible GPS satellites on March 2, 2010 at Danish GPS Center, Aalborg, Denmark.
We consider a case with a slowly moving receiver, that is a velocity a few m/s and clock frequency drift $f_d$. The linearized observation equations are [unit: m/s]:

$$
\begin{equation}
\mathbf{b} = \begin{bmatrix}
doppler_{\text{obs}}^i - \doppler_{\text{comp}}^i \\
\vdots \\
\doppler_{\text{obs}}^i - \doppler_{\text{comp}}^i \\
\vdots \\
doppler_{\text{obs}}^i - \doppler_{\text{comp}}^i \\
\end{bmatrix} = A \mathbf{x} + \mathbf{e} = \begin{bmatrix}
\dot{\mathbf{e}}^i \\
\vdots \\
\dot{\mathbf{e}}^i \\
\vdots \\
\dot{\mathbf{e}}^i \\
\end{bmatrix} \begin{bmatrix}
x \\
y \\
z \\
f_d \\
t_c \\
\end{bmatrix} + \mathbf{e}.
\end{equation}
$$

We quote the observation equations for a dynamic receiver—without derivation [m/s]:

3rd Sogei Workshop, January 21–22  © 2016 by Kai Borre
The matrix $A$ has as many rows as tracked satellites. The system is solved by least-squares method and we get position and time—with initial time error of hours, see the table on the next slide.
## Development in Snap-shot Technique

<table>
<thead>
<tr>
<th>Author</th>
<th>Initial time error of minutes</th>
<th>Correction Initial position of integer ms rollover</th>
<th>Initial time error of hours</th>
</tr>
</thead>
<tbody>
<tr>
<td>Peterson (1995)</td>
<td>x</td>
<td></td>
<td></td>
</tr>
<tr>
<td>van Diggelen (2000, 2009)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
<tr>
<td>Fernández-Hernández (2015)</td>
<td>x</td>
<td>x</td>
<td>x</td>
</tr>
</tbody>
</table>
Numerical Results

We quote the standard deviation $\sigma$ available by the method. We use data from five different locations with 80 samples each of 10 ms (except location 2 which contains 100 samples). The sampling frequency is 16.368 MHz and the IF 4.129 945 MHz (except location 2: 38.192 MHz, 9.548 MHz). The initial receiver time is 1 day off. On a laptop with a standard Intel i7-30QM CPU @ 2.4 GHz the computation takes 0.4 s. The specialized Matlab receiver acquires signals coherently for 1 ms with 10 non-coherent integrations for a total of 10 ms. We use code phase parallel acquisition which is described in Borre et al. (2007).

The doppler and code phase observations were generated in open-loop mode, that is, without tracking loops. The code phase accuracy is limited by the sampling frequency.
The method allows to solve time errors of weeks, months, et cetera. *That is no time reference whatsoever is needed.*

<table>
<thead>
<tr>
<th>Location</th>
<th>Date</th>
<th>2D-σ [m]</th>
<th>3D-σ [m]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Aalborg, Denmark</td>
<td>March 2, 2010</td>
<td>5.6</td>
<td>11.4</td>
</tr>
<tr>
<td>Boulder, USA</td>
<td>May 7, 2005</td>
<td>6.0</td>
<td>8.2</td>
</tr>
<tr>
<td>Barcelona, Spain</td>
<td>March 25, 2010</td>
<td>12.3</td>
<td>35.2</td>
</tr>
<tr>
<td>Barcelona, Spain</td>
<td>June 17, 2010</td>
<td>6.8</td>
<td>34.4</td>
</tr>
<tr>
<td>Barcelona, Spain</td>
<td>February 12, 2014</td>
<td>6.4</td>
<td>11.8</td>
</tr>
</tbody>
</table>

Our results are not especially accurate, as this was not the focus of the research. They can be made much more accurate through state-of-the-art methods.
Time and Position Errors Versus Doppler Errors

Time error as determined from course time doppler solution versus doppler $\sigma$ for 80 snapshots, similarly for position errors.

The gist of the above plots is to demonstrate that in case the doppler observations are better than 20 Hz the method always works. The 20 Hz is achieved within a few ms of processing.

3rd Sogei Workshop, January 21–22  © 2016 by Kai Borre
Cold Snap-shot 2D-positions at Aalborg, Denmark

Positions computed using the cold snap-shot formulas for data from March 2, 2010 at Danish GPS Center, Aalborg, Denmark.

3rd Sogei Workshop, January 21–22  © 2016 by Kai Borre
Isn’t the Idea of a Snap-shot Receiver Only a Theoretical Problem?

Will not all receivers—including snap-shot receivers—not always be synchronised within 1 minute?

Imagine ultra-low-power devices that do not have a running clock and which only are powered on at certain events and without any knowledge of time or location. Examples may be tagging of objects (containers), tracking of animals (migratory birds), et cetera.
Vectorized Receivers
Separate Tracking Loops

Traditional GPS receivers employ separate tracking loops for the individual satellites. Typically a delay lock loop (DLL) is used to track the code sequence (PRN), and another loop is used to track the carrier part of the signal. This tracking operate well under good carrier power-to-noise ratio and low dynamics. Tracking loops track every satellite independently so lock on one satellite does not help the tracking of another satellite.
The loops CH1–CHn are closed inside each channel. Only the pseudorange and pseudorange rate observations are fed to the navigation processor which often is an *iterative least-squares algorithm*. 
Vector delay lock loops (VDLL) make an attractive tool that can provide tracking in very weak signal environment. In VDLL all channels are processed together in one algorithm which typically is an extended Kalman filter. Therefore, even if signals from some satellites are very weak the receiver can track them based on filtered output from the other satellites.
The components of the state vector are the receiver position and the receiver clock offset. The extended Kalman filter controls the NCO. A fast change of the doppler frequency may limit the performance of a vector tracking loop.

As always, a Kalman filter is sensitive to gross errors: an error in one channel immediately propagates into the remaining channels. This does not happen in the separated channel receiver.
The left panel shows the behavior of separated tracking when the signal power from one satellite is rapidly decreased. The separated DLL fails to reacquire the signal when it reappears after 10 s.

The right panel shows results from the vector processor. When $C/N_0$ falls below a certain threshold, the diagonal terms of the noise covariance matrix $\Sigma_e$ are inflated and the vector processor algorithm starts ignoring the observations. During the signal attenuation period the vector processor ignores the code phase observations from the satellite and it uses its own estimates of code phase error. The output of the code discriminator is dominated by noise during the attenuation period. After the signal returns, the code discriminator shows valuable output. The vector processor recovers the correlation power instantly as compared to the separated DLL.

The vertical axes in the upper row indicate correlation power to be multiplied with $10^6$. The lower row shows code tracking error in chips. The figures are copied from Kanwal et al. (2010)
Tools for Receiver Development
Developers of software receivers work often with real signals rather than generated signals, see Plaušinaitis et al. (2010). A copy of the received signals is saved in a file on the laptop.
Later the laptop software instructs the debugging tools in the receiver to substitute the digital IF stream from the front-end with data recorded previously. The signal record is played back to the receiver. The ADC sampling clock signal is used to clock the played back IF signal samples.

A controlling hardware must be added to share the same USB chip by the two data streams. The USB chip also must be steered to indicate which data stream sample is currently transferred. The USB connection natively supports multiple data streams in parallel. Note that the first in, first out (FIFO) buffer for the IF signal must support data transfer in both directions if a signal playback function is required. This setup enables repeatable tests with the same recorded GNSS signals.

The received signals can be thoroughly analyzed in post-processing mode to identify the cause of receiver problems or to do detailed studies of the recorded signal.
Acknowledgements

This presentation is based on discussions with and help from Darius Plaušinaitis and Ignacio Fernández-Hernández.
Selected References


New Textbook on Software Receivers

In 2016 Birkhäuser, Boston publishes a new book on GNSS Software Receivers. It comes with a multitude of Matlab receiver codes and with the front-end sdrnav40. The authors include

- Kai Borre
- Stepan Shafrin
- Heidi Kuusniemi
- Ignacio Fernández-Hernández
- Simona Lohan
- Gonzalo Seco Granados
- Jose Lopez-Salcedo
The popular Easy Suite is updated for RINEX version 3.03 and Matlab R2015a. The code will be released in the first half of 2016.