GPS Position Can Be Computed without the Navigation Data

Niilo Sirola, Tampere University of Technology, Tampere, Finland

Jari Syrjärinne, Research and Technology Access, Nokia Mobile Phones, Finland

BIOGRAPHY

Niilo Sirola is a M.Sc. student at Tampere University of Technology majoring in technical mathematics and software design. He has been researching assisted GPS positioning algorithms since 2000.

Jari Syrjärinne received his M.Sc. degree in 1996 and Doctor of Technology degree in 2001 from Tampere University of Technology, Finland. Since 1999, he has been working for Nokia Mobile Phones where he also finished his doctoral thesis about modern techniques for personal positioning. His current research topics include cellular positioning, sensor fusion, and AGPS.

ABSTRACT

This paper presents an algorithm for computing a GPS receiver's position and the current time from C/A-code phase measurements to at least six satellites and a set of valid satellite ephemerides. Reference position and reference time are not necessary. The process uses a cost function having numerous local minima in addition to the global minimum. An exhaustive search over the 5-dimensional time-position-bias search space will reliably find the global minimizer, and thus solve the approximate position and time. Introducing a reference time within one minute from the true time resulted in worst-case computation time of a few seconds. If the current pseudo-range correction parameters are available, the accuracy of this method is comparable to that of the conventional pseudo-range methods. As a conclusion, an accurate GPS position can be solved in a reasonable time even without the navigation data or precise system time.

INTRODUCTION

The method presented in this paper is intended for GPS positioning in weak signal situations where the satellite signals are so noisy that C/A codes can be tracked but the navigation messages are beyond recognition [1,2,3]. The ephemeris and pseudo-range correction parameters could be already in the receiver's memory, or they could be acquired via e.g. cellular network along with a delayed reference time [1]. A local positioning algorithm for this situation that requires initial time and position approximations has already been presented in [4]. This paper describes a global extension to the local algorithm.

BACKGROUND: THE LOCAL ALGORITHM

Use the following symbols:

t, r, b system time, 3D user position and user clock bias to be solved,

\( \phi_i \) C/A code phase measurement to \( i \)th satellite,

\( \epsilon_i \) approximate ranging error compensation,

\( s_i(t) \) position of \( i \)th satellite at system time instant \( t \), and

\( \tau_i \) approximate time-of-travel from \( i \)th satellite to user.

The range fit to the \( i \)th satellite is defined by

\[
q_i(t, r, b) = \frac{\text{round}}{\Lambda} \left( \phi_i + \epsilon_i + b - \frac{s_i(t) - r}{\Lambda} \right)
\]

where \( \Lambda \approx 300 \text{ km} \) is the length of the C/A code sequence and the lambda-fraction operator is defined by

\[
\text{frac}_\Lambda x \triangleq x - \Lambda \text{ round} \frac{x}{\Lambda}.
\]

Note that the user clock bias \( b \) can be restricted between \(-\frac{1}{2}\Lambda \) and \( \frac{1}{2}\Lambda \) since the fraction operator eliminates integer multiples of \( \Lambda \).

Denoting \( x = \begin{bmatrix} t \\ r \\ b \end{bmatrix} \) and \( q(x) = \begin{bmatrix} q_1(x) \\ \vdots \\ q_n(x) \end{bmatrix} \), we can write the sum of squares cost function

\[
f(x) = \frac{1}{2} \left\lVert q(x) \right\rVert^2,
\]

which has a minimum at the true position \( (T, R, B) \).

When \( n > 5 \) and the satellite geometry is non-pathological,
i.e. both satellite positions and velocities are non-coplanar, this minimum is the unique global minimum.

Figure 1 shows an example of the cost function projected into two dimensions. There are various local minima in addition to the global minimum at the true position. If any gradient-based minimization method is initiated far enough from the global minimum, the search will stray into one of the local minima and not be able to find the true position.

$$\begin{align*}
\text{Figure 1: Two-dimensional slice of the cost function}
\end{align*}$$

An initial point \((t_0, r_0, h_0)\) lies "close enough" to the global minimum \((T, R, B)\) at least when

$$\kappa |T - t_0| + \|R - r_0\| + |B - h_0| < \frac{1}{\gamma} \Lambda, \tag{4}$$

where \(\kappa = 710 \text{ m/s} \) (maximum Doppler velocity) [4]. If this condition is satisfied, then the true position can always be reached with just a couple of Gauss-Newton steps given by

$$\begin{align*}
x_{k+1} &= x_k - \left[ \nabla q(x_k) \nabla q(x_k) \right]^{-1} \nabla q(x_k) q(x_k)
\end{align*} \tag{5}$$

GLOBAL ALGORITHM

The local algorithm finds the correct solution given an initial point inside the "attraction basin" of the true solution. Next, we construct a global extension of the local method. The purpose of the global phase is to launch the local search with different starting points, and identify whether the resulting point is the global minimum or not.

Fortunately, it is easy to detect when the search is going to fail, because the cost function is smooth and convex near the true solution. Most often, when started outside the attraction basin, the Gauss-Newton search takes so large a step that it would take the solution outside the region defined by (4). This is a clear indication of failed search.

Also, the cost function value at the true position is considerably smaller than in any of the local minima.

The search space

The search space is five-dimensional, consisting of the system time a.k.a. GPS time, 3D position, and user clock bias. Under the assumption that the satellite ephemeris is valid, the system time is bound within two or three hours from the time-of-ephemeris, depending on whether the ephemeris is fitted for four or six hours. The user is assumed to be on the Earth surface or within couple of kilometers above or below it, and the clock bias range equivalent is by definition inside \(\pm \frac{1}{\gamma} \Lambda\).

The brute-force approach is to span a grid over the search space such that at least one grid point necessarily satisfies the condition (4). If the system time is divided into intervals of two minutes, it suffices to divide the surface of the Earth into squares 100km across and the bias into intervals of 100km. This results in over 15 million distinct points to inspect. We can start the local search from every point and pick the resulting minimum with the smallest cost function value as the solution.

The computational load of the brute-force approach is roughly equal to 15 million point position fixes. This is hardly practical. There are two ways to accelerate the search: reduce the number of grid points, and speed up the local search.

Satellite visibility

Instead of searching the whole Earth, we can restrict the user position to the area where all the measured satellites are above the horizon. Figure 2 demonstrates the visibility region of three satellites. The point \(\mathbf{r}\) on the Earth surface belongs to the visibility area roughly when

$$\begin{align*}
\mathbf{s} - \mathbf{r} \geq r_{\text{Earth}}^2 \quad \forall i
\end{align*} \tag{6}$$

$$\begin{align*}
\text{Figure 2: Satellite visibility region}
\end{align*}$$

In case of eight satellites, the visibility region has an average area of 30 million square kilometers, which is
about 5% of the total Earth surface area. The more satellites are available, the smaller the visibility area, and thus less grid points to search. This suggests that all the available satellites should be used in this visibility search phase, even if some of those have such a low signal-to-noise ratio that they should not be used in the actual position solution.

In practice, it is most efficient to divide the four-hour system time span into intervals of about 2 minutes and search the whole visibility region of one interval before moving to the next one.

**Gradual local search**

Secondly, we try to reduce the time spent inspecting one point. Instead of executing the full local search at every point, we try some heuristic methods to spot and eliminate the "bad" points as fast as possible, leaving us hopefully with just one point that satisfies all the conditions and is the true solution.

First, given time and position, we check if there is a bias value such that all range fits $q_i$ have absolute value smaller than some pre-defined constant $q_{\text{thres}}$. The denser the search grid is, the lower value can be set for $q_{\text{thres}}$. However, if $q_{\text{thres}}$ is too small the search might miss the true position.

If a good initial bias was found, we launch a rough local search from the obtained point. This phase uses roughly approximated cost function and takes just a few iteration steps to check whether the search converges or not. The search is terminated if the iteration step is too large and would take the solution outside of the assumed attraction basin or too far from the Earth surface.

Rough search initiated inside the attraction basin of the true solution will produce a position estimate with an accuracy of a few kilometers. However, a few additional local minima are usually found.

In the final phase, the full-precision local search is launched starting from the coarse minimum. Some of the searches fail to converge, but most of them produce a local minimizer of the cost function. Fortunately, the cost function value at the global minimum is considerably smaller than in any of the local minima. If $f$ is below a threshold value calculated from the chi-square distribution, we have found the true position, and stop the search.

**Reference position or time**

It is also possible to use any kind of additional position or time information to further restrict the search space, e.g. cell coverage if cellular base station coordinated are available. Usual cell ranges are within few tens of kilometers in rural area and few kilometers in more urban areas.

Wireless assistance can also include a coarse position estimate, which naturally can also be used. In this case, the position search space reduces to just one point and only the correct time has to be found. Conversely, roughly known system time restricts the system time search to just one two-minute interval.

**TESTING**

The proposed methods were implemented in Matlab for testing and simulations. Actual ephemeris from the morning of May 16th 2001 was used, but all the measurements were simulated.

Figure 3 depicts a user in Tampere, Finland using satellites 1, 6, 10, 17, 24, and 30 on May 16th 2001 at 6:12 UTC. The figure shows the edge of the visibility region of the satellites at the time and the points for which the initial bias was found as red circles. The rough local search from these points produced six solution candidates, marked with blue crosses. Only one point, the actual user position, satisfied the cost function threshold after the refined local search, and it is marked in the figure as a blue asterisk.

![Figure 3: Visibility region, position candidates and true position](Background: Day Earth Texture Map ©2002 The Living Earth, Inc.)

We also studied the time Matlab spent computing a single position fix. While this does not predict the actual DSP load, it will give some overall insight of the complexity of the problem. Figure 4 shows statistics about the time-per-fix as a function of satellites used. The more satellites are available, the smaller the satellite visibility region gets and thus the number of possible solutions to check decreases. The position search is terminated as soon as a satisfactory solution is found, which could with good luck happen with very first points checked, or with bad luck the last one. Thus, the total computation times vary from a fraction of a second up to minutes. Overall, it seems that at least eight satellites are required to fix the position in reasonable time. The maximum observed positioning time for seven satellites was 2.6 minutes, and for six satellites over 7 minutes.
Finally, we also experimented with initial time known within one minute. This is a very modest requirement for timing assistance, and even the receiver's internal clock should be able to meet this requirement. With this information, the computation time is reduced by two orders of magnitude (Figure 5).

CONCLUSIONS

As a conclusion, an accurate GPS position can be solved without the navigation data and without accurate time assistance from the network, sometimes even in a reasonable time. The receiver still needs the ephemeris parameters and the pseudo-range correction parameters, but they are valid for hours after reception and thus easy to deliver to the receiver in time. In fact, there already are cellular standards for US CDMA, US TDMA, and European GMS and UMTS systems for network assistance messages containing these information elements [5,6]. The presented method makes it possible to solve GPS position with rather modest assistance.

In comparison to the conventional GPS and AGPS (Assisted GPS) or WAG (Wireless Assisted GPS) solutions, this method has several potential advantages. Most importantly, GPS navigation becomes possible in some bad signal conditions, even with time assistance significantly worse than the 1.5 or 3 seconds required by the earlier solutions [2, 3]. Additionally, while the conventional receiver must listen to at least a few seconds of the navigation message in order to acquire the system time, this algorithm can start calculating as soon as the assistance data is present.

The drawback is, evidently, the increased requirement of computational power at least in the case where even the coarse reference time is unavailable.

When compared to the conventional least squares pseudorange positioning method, range fitting produces somewhat less accurate position fixes due to the system time and thus satellite positions being solved inaccurately [4]. However, the accuracy difference is marginal and negligible in practice since it will be buried in multipath etc. especially in weak signal situations.

Another downside of the presented method is that exact GPS time cannot be solved thus not making it possible to acquire accurate time from GPS. On the other hand, exact time is not needed in navigation applications or in emergency call positioning in which only good coverage and rapid time-to-fix will make a difference.

REFERENCES