Fast and Robust GPS Fix Using One Millisecond of Data

Pascal Bissig
ETH Zurich
bissigp@ethz.ch

Manuel Eichelberger
ETH Zurich
manuelei@ethz.ch

Roger Wattenhofer
ETH Zurich
wattenhofer@ethz.ch

ABSTRACT

GPS is used for outdoor localization in a large variety of applications. Current receivers consume too much power for energy-constrained situations like continuous location tracking on small wearable devices. Mainly, this is due to the large amount of GPS signal that has to be decoded to compute the first position fix. While Coarse-Time Navigation (CTN) can reduce the necessary signal to a few milliseconds, it is not robust to noise. Collective Detection (CD) of satellites can mitigate noise to some degree, but the basic method is computationally expensive. We show how CD can be solved optimally *and* efficiently. Furthermore, we improve the accuracy of CD by exploiting the shape of the likelihood function. All our results are based on real-world signal observations and we achieve localization accuracies of less than 25 meters using a single millisecond of signal. When using 10 consecutive millisecond samples the accuracy improves to less than 10 meters.

CCS CONCEPTS

•Information systems →Global positioning systems;

KEYWORDS

Collective Detection, robust localization, Assisted GPS, low power, maximum likelihood, global optimization

ACM Reference format:
Pascal Bissig, Manuel Eichelberger, and Roger Wattenhofer. 2016. Fast and Robust GPS Fix Using One Millisecond of Data. In *Proceedings of The 16th ACM/IEEE International Conference on Information Processing in Sensor Networks, Pittsburgh, PA USA, April 2017 (IPSN 2017),* 11 pages.
DOI: http://dx.doi.org/10.1145/3055031.3055083

IPSN 2017, Pittsburgh, PA USA

978-1-4503-4890-4/17/04…$$15.00
DOI: http://dx.doi.org/10.1145/3055031.3055083

1 INTRODUCTION

Location sensing has proven to be an important prerequisite for many applications. For example navigation, tracking, life-logging, research such as animal tracking, and rescue services. Many classes of battery powered devices are more useful when position information is available, such as smartphones, cameras, fitness trackers, smart watches and sensor nodes. For most outdoor scenarios, GPS is the localization system of choice, mainly due to its global coverage and accuracy.

However, continuous GPS receiver operation still consumes too much energy for mobile devices such as fitness trackers or even smartphones, since current receivers cannot be efficiently duty-cycled. When the receiver is switched off for a few minutes to conserve power, it takes a lot of time and energy to compute a new position fix once it is turned back on again. This has far-reaching consequences for many application scenarios. For example, today's GPS receivers make us wait for a first fix, which can be annoying if one wants to navigate an unknown place. Also, geo-tagging photos is not instant and energy consuming. Due to the energy consumption issues, many applications, such as long term tracking, are still out of reach.

In this paper, we present a receiver which requires only a single millisecond of GPS signal to compute its position. This means that the signal can be recorded and stored locally for later processing. The signal recording can be sent to a remote server which can perform the energy consuming position computation. This translates to a reduction in power consumption as well as an increase in convenience for many applications. For example, the initial position when navigating with your phone can be found within a few milliseconds depending on network latency. A smartwatch or fitness tracker may be able to track its location every few seconds for weeks at a time. When the duty cycle is further reduced, a tracking device that only requires one position fix per hour may run for years on a single coin cell battery. Geo-tagging photos can be simplified to adding a one ms signal recording to the photo which is stored and the position can be computed later on.

The GPS signal that reaches the surface of the earth is weak due to the path loss. To reduce the effects of noise, current receivers track GPS signals over extended periods of time. Since we want to be able to store the recorded signals for later processing, this is not a feasible solution for us. To still increase the noise tolerance of our approach, our solution yields the position fix that best explains the given signal measurement. This means that we do not need to detect satellite ranges which easily can throw off current GPS receivers as well as CTN receivers.

The problem of finding the position that best explains a given signal measurement is non convex. Hence, the solution cannot be found by iteratively improving a candidate solution in all cases. If the location is approximately known, finding the most likely

position can be achieved by computing the likelihoods of all close-by positions and selecting the most likely one. The more uncertain the initial guess about position and time, the larger the search space (position and time) becomes. Computing all the likelihoods presents a computationally expensive maximization problem in this case. However, we show how the global maximum can be found efficiently using a branch and bound approach. The runtime of the algorithm is correlated with signal quality: In good signal conditions, the computational load is low. The worse the signal conditions become, the higher the computational burden. However, the best position and time fix is found in any case. The branch and bound implementation speeds up the acquisition time and hence also the *time to first fix (TTFF)*.

We exploit the shape of the likelihood function to achieve higher positioning accuracy and robustness. As a result, under similar conditions (signal duration and sampling rate), our method leads to more accurate positioning compared to previous approaches. Furthermore, we show that there is a trade-off between the amount of sampled signal used and the accuracy of the positioning solution. If we average over two consecutive location fixes from one millisecond of data each, the median error is reduced from 25 to 15 meters. Averaging over 30 fixes (0.03 s of signal), the median error is as low as 6 meters. Tracking a user's position decreases the computational complexity of each consecutive fix as the search space (space and time) is much smaller.

2 RELATED WORK

Van Diggelen [9] has introduced the idea of *Coarse Time Navigation (CTN)*. Using CTN, a position fix can be found from only a few milliseconds of data without decoding any data from the GPS signal. The requirement for this is prior knowledge of the receiver time and position to within a few seconds and 150 kilometers, respectively. Liu et al. [6] showed that since CTN only requires a few milliseconds of data, the raw signal can be stored and the computation can be outsourced or postponed until power is available. This mitigates the problem of high energy consumption for acquisition by not acquiring the satellites on the receiver, enabling duty cycling. However, due to the short signal duration, accuracy and robustness is worse than in classic receiver designs relying on acquisition and tracking stages. Our GPS receiver design extends this idea and can compute a position from a *single* millisecond of signal. Our localization method counteracts the effect of the short signal duration and improves positioning accuracy compared to existing work on CTN. Also, we show how accurate position fixes can be computed from inaccurate time estimates. This allows us to drop the heavy and power consuming DCF-77 clock receiver required by Liu et al. [6]. As a result, our receiver can be miniaturized and can function for years even when there is no clock synchronization except at the very beginning.

A second branch of research is concerned with improving the robustness of GPS receivers. In classical GPS, the receiver location is determined based on signal parameters. The most important ones being Doppler shift and code delay for each satellite. From these parameters, a position in space is computed. Clearly, signal parameters may be erroneously detected which leads to unusable position estimates.

Instead of estimating the signal parameters, Closas et al. [4] showed how the receiver position can be estimated directly and how this can improve the robustness of GPS receivers.

We refer to the basic idea as *Collective Detection (CD)*, but it is also called *Direct Positioning* or *Combined Detection* in the literature. Evaluations of CD have been performed in both simulation and practice [2–4]. The main concern is the computational complexity introduced by the high-dimensional search space. Also, the likelihood function is generally non convex, prohibiting standard greedy maximization methods. Optimizations such as the one proposed by Axelrad et al. [2] reduce the computational complexity but cannot guarantee that the best possible position is found. We improve the robustness of our approach by applying CD. Especially so in multipath environments because CD finds the globally best solution whereas classical receiver designs depend on correct pseudorange estimates for each individual satellite. Hence, one bad pseudorange estimate can throw off the classical solution whereas the most likely position (in CD) may still remain unaffected. However, CD is expensive in terms of computation. To alleviate this drawback of CD we introduce a branch and bound algorithm which yields reduced computational complexity while still guaranteeing that the best possible solution is found.

3 GPS FUNDAMENTALS

The GPS system conceptually consists of three parts: the control segment, the space segment and the user segment. The space segment nominally consists of 24 satellites orbiting the Earth [5]. A network of monitor stations and ground antennas makes up the control segment. It is primarily used to monitor the satellites state and keep track of their ever changing orbits. The orbits need to be known as accurately as possible to improve localization accuracy [5][1]The third – and for our discussion most important – part of GPS are the receivers, making up the user segment.

3.1 GPS Signal

The satellites transmit signals in different frequency bands. These include at least the so-called L1 and L2 frequency bands at 1.57542 GHz and 1.2276 GHz [5]. The signals are transmitted through a helix array antenna which right-hand circularly polarizes the signals [5]. This helps suppressing multipath signals at a receiver because a reflection of the signal polarizes it in the opposite direction.
In order to distinguish the signals from different satellites and to extract the signals from the background noise, code division multiple access (CDMA) is used.

Figure 1 shows the modulation scheme utilized in GPS. The Coarse/Acquisition code (C/A code) is a sequence of 1023 bits which is unique for each satellite. Specifically, Gold codes are used to achieve favorable correlation and cross-correlation properties [9]. Because Gold codes look like random bit strings, C/A codes are also called pseudo-random noise (PRN) sequences. The C/A code is transmitted at 10.23 MHz which means it repeats every millisecond. The data is transmitted at $50\frac{\text{bit}}{\text{s}}$ and hence, each bit contains 20 complete C/A cycles. The data and C/A code are merged using an XOR before being mixed with the L1 or L2 carrier. Figure 1 shows how the GPS signal is generated. Note that for better readability,

[1]Further information can be found at http://www.gps.gov/systems/gps/control/

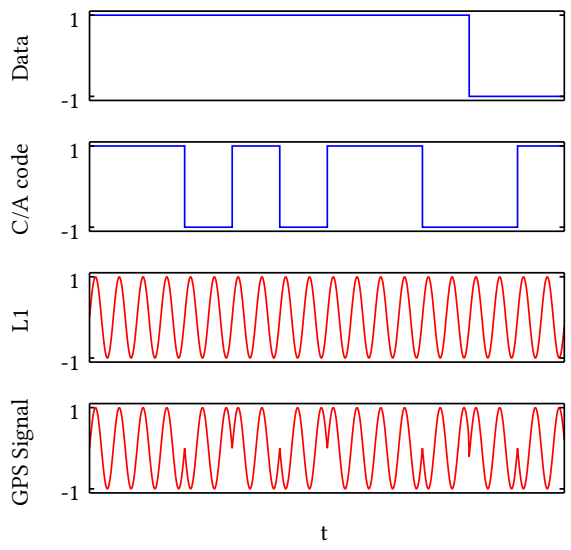


Figure 1: The structure and modulation of the GPS Signal. The binary data and C/A code are mixed with the carrier frequency (L1) using the BPSK modulation scheme.

the C/A frequency and the L1 frequency do not have the correct ratio. The data that is broadcast contains a timestamp (called HOW) which can be used to compute the location of the satellite when the packet was transmitted. However, to do this, the receiver needs accurate orbital information (called ephemeris) about the satellite which changes over time. While the HOW timestamp is broadcast every six seconds, the ephemeris data can only be received if the receiver can decode at least 30 seconds of signal.

3.2 Localization

Conventional GPS receivers use three stages when obtaining a location fix.

Acquisition. First, the set of available satellites has to be found. This can be achieved by correlating the received signal with the known C/A codes from the satellites. Since the satellites move at considerable speeds, the signal frequency is affected by a Doppler shift. Hence, receivers usually correlate the received signal with C/A codes with different Doppler shifts.

Tracking. After a set of satellites has been acquired, the data contained in the broadcast signal is decoded. Doppler shifts and C/A code phase are tracked using tracking loops. After the receiver obtained the ephemeris data and HOW timestamps from at least four satellites, it can start to compute its location.

Localization. Localization in GPS is achieved by multilateration, a technique using *time difference of arrival (TDOA)* measurements to compute a position. The arrival times of the HOW timestamps received in the tracking phase are used to compute the set of TDOAs.

In *trilateration*, the distances between a mobile station and some stations with known position are measured. This can for instance be done through *time of arrival (TOA)* measurements, when the signal transmission times are known and all the stations are synchronized in time. The position of the mobile station lies at the intersection of the spheres around the stations with fixed position with the corresponding radii.

While the satellites operate on an atomic frequency standard, the receivers are not synchronized to the GPS time. Therefore, the local time at a receiver is unknown. Due to that, the distance of the receiver to the satellites cannot be directly computed from the local arrival time of the signals at the receiver. Instead, only the time *differences* of the arrival times can be measured. Therefore, the multilateration method is applied. From a computational view, there is not a large difference between the trilateration and the multilateration approach. The latter problem just contains one more variable, which is the receiver time, and hence needs measurements from at least four instead of three satellites for the problem to be well-defined. The receiver position is found through a least squares optimization.

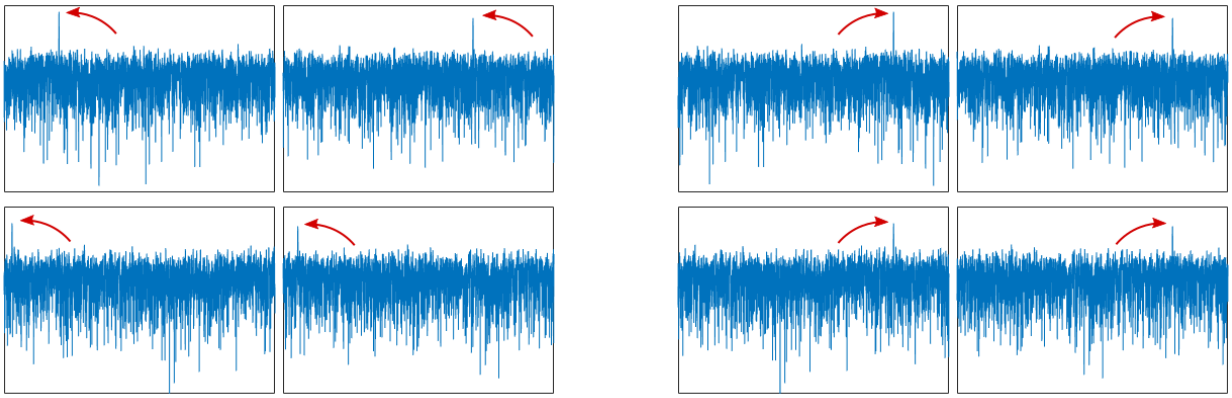
3.3 Assisted GPS

One of the main disadvantages of GPS is the low bit rate of the navigation data encoded in the signals transmitted by the satellites. The minimal data necessary to compute a position fix, which includes the ephemerides of the respective satellites repeats only every 30 seconds. In order to decode all that data, the receiver has to continuously track and process the satellite signals which induces a high energy consumption. Furthermore, upon starting up a receiver, a position will not be instantly available. To overcome this drawback, receivers can run continuously, but this consumes even more power.

Assisted GPS (A-GPS) drastically reduces the startup time by fetching the navigation data over the Internet, commonly by connecting via a cellular network. Data transmission over cellular networks is faster than decoding the GPS signals and normally only takes a few seconds. Using this data, the acquisition time can be reduced since the set of available satellites can be estimated along with their expected Doppler shift. This is possible because the exact arrival time of the navigation data is not required for the localization. Also, ephemeris data is valid for at least 30 minutes. However, the receiver still needs to extract the HOW timestamps from the signal but since they are transmitted every six seconds, this is roughly how much time it takes an A-GPS receiver to compute a position fix.

3.4 Coarse-Time Navigation

Coarse-Time Navigation (CTN) is an A-GPS technique which drops the requirement to decode the HOW timestamps from the GPS signals. Van Diggelen [9] describes the concept in detail. The only information used from the GPS signals are the phases of the C/A code sequences which are detected using a matched filter. These arrival times are directly related to the sub-millisecond parts of the corresponding TDOAs. The number of whole milliseconds of the signal propagation time are resolved with a known approximate location and time. Because the signals travel at the speed of light, which is about 300 km per millisecond, in order to be able to resolve the number of whole milliseconds unambiguously, the deviation



(a) Original (not shifted).

(b) Shifted (circularly) according to the distance from the receiver to the corresponding satellite.

Figure 2: Correlation functions for four satellites. On the left there are the correlations of the received signal with the PRN sequences of four different satellites. The spikes indicating the beginning of the PRN codes in the received signal are marked with red arrows. If we shift the correlation vectors according to the true distance to the satellites, we see on the right hand side that the peaks all align.

may at most be 150 km from the correct values. Here, the deviation is defined as the time offset multiplied by the speed of light plus the position distance. Since the PRN sequences repeat every millisecond, without considering navigation data bit flips in the signal, CTN can in theory compute a position from one millisecond of the sampled signal. But since bit flips can happen, to make sure all visible satellites can be used, two milliseconds are necessary. With such short signal recordings, clearly noise becomes a major issue, because noise cannot be filtered out as easily as with much longer recordings of several seconds. But the advantage of this extremely short recording period is that the signal processing is fast and power-efficient and thus also the latency of a first fix. Also, since no metadata has to be extracted from the GPS signal, CTN may be able to compute a location even if the GPS signal cannot be decoded anymore due to noise or attenuation.

3.5 Collective Detection

Collective detection builds upon the observation that detecting peaks in the correlation functions of individual satellites might yield sets of pseudoranges which are not consistent with the laws of physics. By searching a solution in space and time directly, this can be avoided. The problem then consists of finding the most likely position given the received signal. From a given hypothetical position and time (referred to as hypothesis in the following), the corresponding ranges of the satellites can be inferred, which are then used to determine the arrival times of the signals. Figure 2 shows how the correlation functions of the received signal with PRN codes of different satellites on the left. On the right, the same correlation functions are circularly shifted by the expected time difference of arrival at the correct location. Clearly, the correlation peaks of all four satellites align. A receiver can exploit this by combining corresponding correlation values from all the satellites to compute a likelihood measure. This is essentially what our receiver does. Erroneous peaks in the correlation function most likely never align which improves noise resistance. Commonly, the hypothesis pseudo-likelihood is defined as the sum of the satellite pseudo-likelihoods, but one could also use other measures, for instance the product.

4 LOCALIZATION METHOD

The basic idea of our method is to asses the quality of many hypothetical receiver states $h = (h^p, h^t)$ which consist of the receiver position h^p and time h^t. The quality of a hypothesis is determined through a likelihood function which assigns a pseudo-likelihood to the hypothesis given external information and the observed signal. This likelihood $\mathcal{L}(h)$ is a measure of how well the observed signal matches the signal expected at a hypothesis h.

4.1 Likelihood

Given a hypothesis h, we can use the knowledge about the satellites' signal transmit times and orbits (from the navigation data) to compute the expected signal phase $\phi_i(h)$ arriving at the receiver from the i^{th} satellite. This is discussed in detail in Section 4.2. Hence, for any hypothesis h we can expect a C/A code with phase $\phi_i(h)$ from satellite i in the arriving signal. We can check how well the received signal $r(t)$ matches this expectation by computing a single correlation value with satellite i's C/A code $\text{ca}_i(t)$.

$$c_i(h) = \sum_{\tau=0}^{1\text{ms}} |r(\tau) \cdot \text{ca}_i(\tau - \phi_i(h))| \tag{1}$$

If our hypothesis h is correct, we expect large correlation values c_i for satellites whose signal can be received, because the code phase of the C/A code in the received signal match the expected code phase $\phi_i(h)$. For satellites that are heavily attenuated or reflected, c_i will be almost completely random. We define our likelihood

function as the sum of the correlation values for a given hypothesis over all *visible* satellites, whose indices are denoted by the set V.

$$\mathcal{L}(h) = \sum_{i \in V} c_i(h) \tag{2}$$

The receiver position and time are estimated by selecting the hypothesis h^* which maximizes the likelihood measure:

$$h^* = \arg\max_{h \in F} \mathcal{L}(h)$$

where F is a set of *feasible* (position, time) tuples.

4.2 Computing the C/A Code Phase

To compute the likelihood of a hypothesis h, we need to know the C/A code phases $\phi_i(h)$ of the visible satellites. In the following, we assume that the signal propagation delay $d_i(h)$ is mainly determined by the distance between receiver and satellite. Note that the maximum signal propagation delay to a receiver on Earth is 87 ms [8]. During this time, a receiver's movement does not have a significant effect on the propagation delay. However, the much faster satellite movement has. Therefore, we compute the propagation delay at the transmit time t_i of a signal even though the receiver may still travel for an additional 87 ms.

The code phase $\phi_i(h)$ relates to the transmit time $t_i(h)$ of the received as follows:

$$\phi_i = t_i(h) \bmod 1 \text{ ms}$$

The transmit time $t_i(h)$ of the received signal at time h^t are related by the propagation delay $d_i(h)$ between the hypothetical position and the satellite position.

$$t_i(h) = h^t - d_i(h) \tag{3}$$

The propagation delay can be found by dividing the spatial distance between the hypothetical position h^p and the satellite position p_i by the speed of light C:

$$d_i(h) = \frac{||h^p - p_i(t_i(h))||}{C} \tag{4}$$

The propagation delay $d_i(h)$ depends on the distance between the satellite position p_i at the time of transmission $t_i(h)$ and the hypothetical position h^p. The satellite position $p_i(t_i(h))$ at a given time can be computed from the ephemeris.

So the propagation delay $d_i(h)$ can be found knowing the transmit time $t_i(h)$ which itself can be found knowing the position of the satellite $p_i(t_i(h))$ which can only be found knowing the transmission time $t_i(h)$ for which the propagation delay $d_i(h)$ needs to be known. This circular dependency can be resolved by a short fixed point iteration which exploits the difference between the speed of light and the speed that the satellites travel with.

Namely, the signal propagation times from a satellite to a receiver on Earth range between 67 and 86 ms [8]. If we compute the signal transmit time using Equation 3 and this crude estimate we get $t_i \approx h^t - (67 + 86)/2$ ms $\approx h^t - 76.5$ ms. The estimation error in the transmit time $t_i(h)$ is at most 9.5 ms. The maximum satellite speed relative to a receiver on Earth is 929 m/s [8]. This means that our estimate for $t_i(h)$ of 9.5 ms leads to a worst case satellite position estimation error of 9.5 ms · 929 m/s = 8.83 m. Using this new satellite position error, the second iteration starts with a new estimate of the transmit time $t_i(h)$, based on a satellite position error which is at most 8.83 m. Hence, the propagation delay estimation error is at most 8.83 m/C = 19.4 ns. The satellite position estimate that can be achieved using this propagation delay estimate already has a negligible error of 19.4 ns · 929 m/s = 18 µm.

4.3 Search Region

To guarantee the uniqueness of the solution, we limit the search region in which the set F of feasible hypotheses is contained. As GPS signals travel at the speed of light C, the C/A code phase of a satellite are the same for two hypotheses if their distances to the satellite differ by $k \cdot C \cdot 1\text{ms} \approx 300\text{km}$ for integer values for k. To avoid this affecting our results, we bound the search region in which the set F of feasible hypotheses is contained to a diameter of 300 km. Most likely the correct solution can still be found in larger areas, especially when more than four satellites are visible. Note that the correspondence between time error and range error is given by the maximum relative satellite speed against a receiver, which is less than $1\frac{\text{km}}{\text{s}}$ on the Earth surface [8]. For instance, a position range of 100 km and a time range of 50 km / $1\frac{\text{km}}{\text{s}}$ = 50 s are guaranteed to deliver a unique solution.

For bounding the solution domain, one can use the antenna position of a cellular network as a reference. When the signal of the satellites is strong enough, we can also find the approximate receiver location with an idea presented by Liu et al. [6]. The authors show how the measured Doppler shift of a signal limits the receiver position to a cone. The receiver positions is then at the intersection of the cones from each satellite. If we do not compute an initial fix, we can use the last computed position as an approximation for the new position.

4.4 Visible Satellites

The set V contains the indices of all potentially visible satellites. It is assumed to be the same for all hypotheses $h \in F$ and is determined as all the satellites with an elevation above the horizon larger than five degrees, as seen from the center of the search region. In theory, V is a function of a hypothesis h and the ephemerides, from which the elevation angles can be computed. However, it is safe to assume V is fixed with respect to all hypotheses since the elevation angles barely change within the search regions we consider. Also the Earth's rotation during the signal transmission can be neglected when computing the elevation angle of a satellite.

4.5 Space Discretization

The computation of the correlation values given in Equation 1 shifts the locally generated C/A code by its expected phase. In our case, the expected phase is rounded such that we shift by an integer value corresponding to one sampling interval T_s of the receiver. This helps to simplify the computation of the likelihood function as no signal interpolation is required. Due to the rounding, the likelihood of two hypotheses that are close may lead to the exact same set of C/A code phases ϕ_i for all visible satellites.

Ideally, we spread hypotheses in the search range such that no two hypotheses correspond to the same set of C/A codes to conserve computation resources. Also, we would like to have one hypothesis

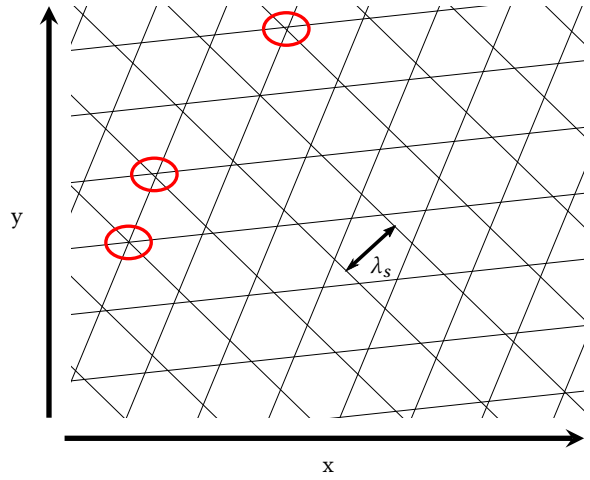


Figure 3: Two dimensional example of search space discretization. In this example, three satellites are visible which cause three groups of parallel lines slicing the search space. When crossing a line, the expected C/A code phase ϕ_i for the corresponding satellite is rounded to the previous or next sampling period T_s. All positions inside a bounded area have the same likelihood. Very small regions may exist (indicated by the red circles).

for every set of C/A code ranges which can be achieved within the search region.

Depending on the sampling interval T_s, we can compute the range difference that is required to change the value of the rounded C/A code phase ϕ_i. Namely, the corresponding "length" of a sample is $\lambda_s = T_s \cdot C$, where C is the speed of light ($\lambda_s \approx 37$ m for $T_s = \frac{1}{8\text{MHz}}$). Thus, for each satellite, the solution space is sliced into *spherical shells* with a slice width of λ_s. Each hypothesis in such a slice produces the same rounded expected C/A code phase ϕ_i.

With multiple satellites, the space is sliced in several directions as shown in Figure 3. This divides the solution space into volumes in which all the hypotheses correspond to the same rounded C/A code phase and therefore equal likelihoods. Figure 3 shows a two-dimensional example.

Since we do not know the exact shape of the division of the search space in the volumes of equal observations, we sample the space with a regular grid. Ideally, this grid would be dense enough to "capture" all these volumes. However, some of these volumes can be infinitely small and thus, with any fixed grid density, we might not sample some volumes and therefore not find the most likely hypothesis.

This means that we cannot guarantee that we sample the volume which corresponds to the highest likelihood that is achievable given the observations. Luckily we can make sure that we do not miss the correct solution completely because no hypothesis is close enough by selecting the grid such that neighboring points are λ_s apart.

In this case, each hypothesis represents a cube of side length lambda. Such a cube has a diameter of $\sqrt{3} \cdot \lambda_s \approx 1.7 \cdot \lambda_s < 2 \cdot \lambda_s$. Since the space is divided into those cubes, an uncovered area can at most be half a diameter apart from the nearest hypothesis, that is the distance to the nearest hypothesis is less than λ_s. Note that a distance smaller than λ_s can at most cross one slice boundary for each satellite. This means that for an uncovered area and its nearest hypothesis the expected code phases ϕ_i are at most one sampling interval T_s apart.

The key observation is that our (and also common) GPS receivers oversample the GPS signals. For the correlation, this means that the peaks are not confined to a single sample of length T_s, rather their neighboring values are quite high as well and form a triangle-like pattern. Without noise, the correlation values at a distance of k samples from the peak have a value of at least $(1 - k \cdot 2 \cdot f_{\text{PRN}}/f_s)$ times the value of the peak. f_{PRN} is the rate of the PRN sequences (1.023 MHz). f_{PRN}/f_s is the fraction of the locally generated PRN sequence which does not match the correct part of the PRN in the signal. For a sampling rate of 8 MHz ($f_s = \frac{1}{T_s}$) for instance, the directly neighboring values of the peak are at least 74 % as high as the peak itself, for a sampling rate of 56 MHz at least 96 %. Assuming the used sampling rate is at least 8 MHz, the found correlation values may at most be 26 % smaller compared to the largest one. Alternatively, we could filter the correlation values such that the correlation value at an index contains the highest correlation values amongst its direct neighbors. In this case, we are guaranteed to find the highest achievable likelihood, but the likelihood function is less sharp. The trade-off that we make here is a decrease in the likelihood at the correct position.

4.6 Time Discretization

The hypotheses also have to be spread in the time domain. As in the spatial discretization above, we have to make sure that we sample densely enough, such that we do not miss the most likely position. If the hypothetical time for the correct location h^p is off by as few as $10 \cdot T_s$, its likelihood will be completely random (assuming $T_s = 8$MHz). This follows from the same argument about the shape of the PRN autocorrelation function above. In order to allow for more coarse sampling in the time domain, we exploit the fact that the expected C/A code phase $\phi_i(h)$ is approximately constant when varying the hypothetical time h^t by less than one ms. Hence, we simplify the computation of the correlation values $c_i(h)$ for hypotheses that are identical up to a difference in time t_μ which is smaller than 1 ms.

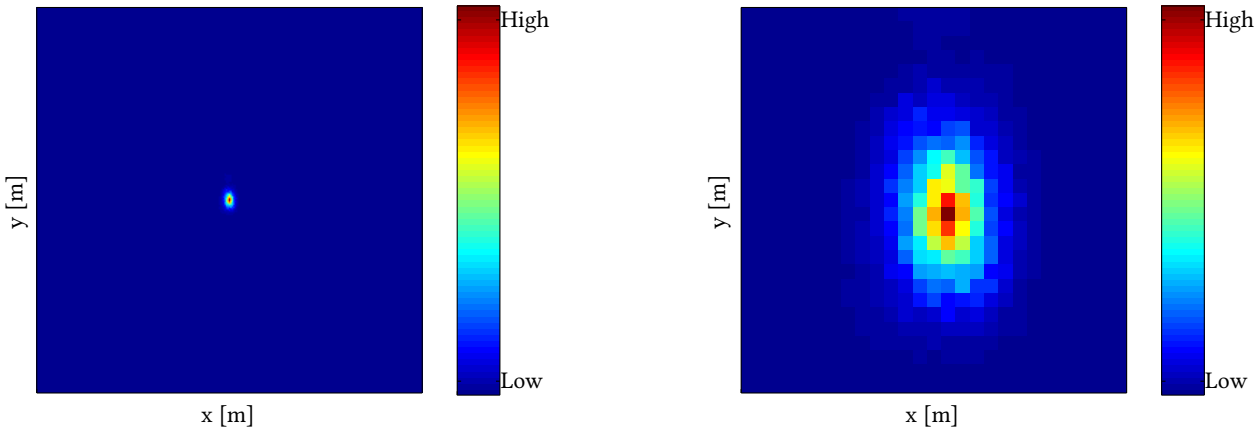
$$c_i(h, t_\mu) = \sum_{\tau=0}^{1\text{ms}} |r(\tau) \cdot \text{ca}_i(\tau - \phi_i(h) - t_\mu)| \tag{5}$$

We can simplify the computation of $c_i(h, t_\mu)$ for all $t_\mu \in [0, T_s, 2 \cdot T_s, \ldots, 1\text{ms}]$ using the correlation function C_i:

$$C_i(t_\mu) = \sum_{\tau=0}^{1\text{ms}} |r(\tau) \cdot \text{ca}_i(\tau - t_\mu)| \tag{6}$$

Note that the correlation function $C_i(t_\mu)$ can be computed independent of the hypothesis. By shifting the correlation function $C_i(t_\mu)$ of the received signal with the C/A code according to the expected phase $\phi_i(h)$, we can simplify the computation of the likelihood as follows:

$$\mathcal{L}(h) = \max_{t_\mu} \sum_{i \in V} C_i(t_\mu - \phi_i) \tag{7}$$



(a) $\mathcal{L}(h)$ on a grid of 10 km by 10 km for a fixed time

(b) $\mathcal{L}(h)$ on a grid of 1 km by 1 km for a fixed time

Figure 4: In situations with line of sight between receiver and satellites, the likelihood function is smooth and unambiguous. The figures shown are a cut through the search space where the time and height of the receiver have been fixed at the values corresponding to the most likely hypothesis. The distance between two points in the grid is approximately 37 meters.

This allows us to choose the time domain to be sampled at up to 1 ms intervals without leaving a good solution undetected.

In the worst case, an inaccurate time hypothesis shifts the most likely position by the maximal speed of the satellites relative to the earth's surface ($1\frac{\text{km}}{\text{s}}$). This means that the localization error is expected to increase less than 1 m if the hypothetical time is off by 1 ms. Hence, we can further increase the intervals at which the time domain is sampled. This does not negatively affect the observations about the spatial discretization. We are still guaranteed to observe hypotheses that are very close to the highest achievable likelihood.

4.7 Averaging Over Likely Hypotheses

So far, we only discussed choosing the hypothesis with the largest likelihood as the solution. As described in Section 4.5, hypotheses that are near the correct solution should get a high likelihood as well, because the PRN is oversampled and therefore its auto-correlation function has a triangular shape around the peak. To improve localization accuracy, we consider the set of hypotheses H with the highest likelihoods. The set of most likely hypotheses is then combined using a weighted average.

$$\bar{h^p} = \sum_{h \in H} \mathcal{L}(h) \cdot h^p$$

In Section 5 we discuss the performance impact of the averaging as opposed to only selecting the most likely hypothesis.

4.8 Efficient Implementation with Branch-and-Bound

Figure 4 shows horizontal cuts of example distributions of our likelihood computed from a one millisecond window of samples in good signal conditions. Our branch and bound method exploits this shape of the likelihood function under clear signal conditions, avoiding the computation of all likelihoods in the search space. The search space as discussed in Sections 4.3 and 4.5 is large as the hypotheses are spread at a distance of 37 m from each other and the search space spans 200 km × 200 km × 30 km. In addition to this, the time domain is searched within 10 s at intervals of 40 ms. This means that there are roughly $2 \cdot 10^{12}$ hypotheses which need to be tested. To reduce the number of hypotheses for which we need to compute the likelihood, we employ a branch and bound method as described in Algorithm 1. To do so, we need a method

Algorithm 1 Finding the n most likely points given a search space defined by a hypothesis h.

```
procedure S = GetMostLikelyPoints(n,h)
    n: the number of likely points contained in S.
    h: the initial hypothesis defining the search space.
    h_lmax = maxLikelihood( h )
    queue.add(h)
    S = ∅
    while queue.hasElement() do
        h = queue.popMostLikely()
        if h_lmax ≤ min_n(h_lmin ∈ S) then
            continue
        end if
        h_lmin = likelihood( h )
        h_lmax = maxLikelihood( h )
        S.add(h)
        h[1] ...h[16] = splitHypothesis( h )
        for h[i] = h[1] ... h[16] do
            h[i]_lmax :=h_lmax
            queue.add(h[i])
        end for
    end while
end procedure
```

to compute both an upper- and lower-bound on the achievable likelihood (indicated by h_{lmax} and h_{lmin}) within an area defined by a hypothesis. Note that in the algorithm, a hypothesis h contains the center position in x,y,z, and t and also the size of the search space around it in all dimensions (x,y,z,t). The initial hypothesis covers the entire search space i.e. it extends over 200 km × 200 km × 30 km × 10 s. We approximate the lower bound of achievable likelihoods within an area as the likelihood of the hypothesis itself (likelihood(h) in the Algorithm). For the upper bound (maxLikelihood(h) in the Algorithm), we use the expected code phases ϕ_i along with the size of the area covered by the hypothesis. Note that the larger the area covered by a hypothesis, the larger the uncertainty about the possible code phases ϕ_i. The uncertainty is given by the diagonal of the area covered divided by the speed of light. For a hypothesis with a diagonal of 10 km, the uncertainty is roughly 33 microseconds which corresponds to roughly 270 sample intervals T_s at 8 MHz.

This can efficiently be taken into account when computing the likelihood as described in Equation 7. Instead of utilizing the correlation function as described in Equation 6, we apply a max-filter first.

$$C'_i(t_\mu) = \max_{\tau \in R} C_i(t_\mu) \tag{8}$$

R is the set of possible shifts that can be expected within the region covered by a hypothesis. In the example above with 10 km diagonal, $R = [-16.5\mu s, 16.5\mu s]$. The likelihood computation stays the same as in Equation 7 but using $C'_i(t_\mu)$ instead of $C_i(t_\mu)$. This yields the highest possible likelihood. To further speed up the computation, the max-filtered correlation functions can be precomputed as it is the same for all hypotheses covering areas of the same size.

Hypotheses in the queue are processed according to their maximum achievable likelihood h_{lmax} (popMostLikely()). This is crucial as areas with great potential are explored first, making it more likely that bad areas are not further explored. Each processed hypothesis is split in two in all dimensions (x,y,z,t) which leads to 16 new hypotheses, covering a smaller volume of the search space each. As soon as a hypothesis cannot achieve a higher likelihood than the n best hypotheses already observed, it is not further split up and discarded. The method guarantees that the n most likely points are found as only hypotheses are discarded which cannot possibly achieve a high enough likelihood.

Clearly, the performance of the algorithm depends on the quality of the received signal as the bounds will be more accurate for a smooth likelihood function. We will analyze the performance degradation as the signal quality deteriorates in Section 5.

4.9 Local Oscillator Frequency Bias

In practice, one of the problems we have to deal with is the frequency error of the local oscillator (LO) in the front end. The LO is not only used for the generation of the reference frequency for the frequency down-conversion, but also as the clock of the ADC. Therefore, the LO error influences two parameters. First, the observed frequencies of the signals from the satellites change. Second, the effective sampling rate or the time that passes per sample changes. Akos [1] states that the frequency for the locally generated C/A code should match the actual frequency with an accuracy better than 250 Hz. Otherwise, correlation peaks are hard to find even under good signal conditions. To get an SNR close to the optimum possible, the accuracy of the frequency should be much better.

During the acquisition phase in conventional receivers, the Doppler shift of each satellite is estimated by correlating the received signal with multiple frequency shifted versions of the C/A code. The frequency shifted C/A code which matches the received signal the best is used to estimate the arrival time and also gives information about the sum of the LO offset and the Doppler shift. After the acquisition, the Doppler shift, and hence the LO, is known only approximately to reduce the computational complexity during acquisition. This approach could be replicated in our solution to estimate the LO offset.

Similar to the search performed in classic receivers, we could track the LO offset by computing the C/A code correlation functions for different frequency offsets. Note that since we do compensate for the Doppler shifts using our prior knowledge, we only need to estimate the LO offset instead of the sum of the LO offset and the Doppler shifts of each satellite.

In our test setup described in Section 5, we measured the LO offset initially using the classical GPS approach. We observed that the offset stayed almost constant even over more than a year. Therefore, careful calibration of the LO can reduce the impact of its errors to an extent that is acceptable. Over the course of 1.5 years, all experiments were performed with the same, constant LO offset (+1.9 ppm).

For an oscillator which does not exhibit such a stable frequency offset over a long time, it would be possible to regularly update the frequency error estimate by correlating with a local signal with slightly lower and higher frequency – similar to the early-late tracking of the code phase in traditional receivers – in situations with good SNR. Since the frequency error will not change quickly, a low SNR of the received signal can be tolerated for an extended period of time without significant performance degradation.

5 EVALUATION

For the evaluation of our method, we used an Ettus USRP B200 software radio with a standard GPS patch antenna from Trimble Navigation. Samples were recorded as 8 bit I/Q samples with 8 MHz sampling frequency. We made recordings of several minutes and cut out windows with one millisecond length every 0.999 seconds. We did not choose exactly one second, since bit flips in the navigation signal, which severely degrade the signal quality, can occur every 20 milliseconds. To prevent these to always have an influence on the same satellites' signal, we chose a slightly shorter interval. Samples with 8 bits were used since this is the lowest number of bits supported by the board's driver. However, we expect that the performance does not significantly vary when only 2 bit samples are used, because using 2 bit samples degrades the SNR by only 0.55 dB [8] (Section 6.12).

We used navigation data originally broadcast from the satellites, which we downloaded from NASA's archive of space geodesy data[2] [7]. For the time synchronization, we determined the time of the first sample received from the RF front-end with the Network

[2]We used the "Daily GPS Broadcast Ephemeris Files" data set that can be found at http://cddis.nasa.gov/Data_and_Derived_Products/GNSS/broadcast_ephemeris_data.html

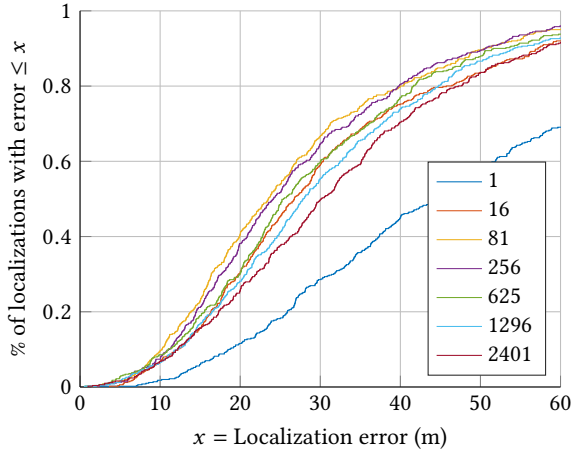


Figure 5: Accuracy of our method with different numbers of most likely points used for the weighted averaging. Cumulative distribution functions of positioning error (distance to ground truth).

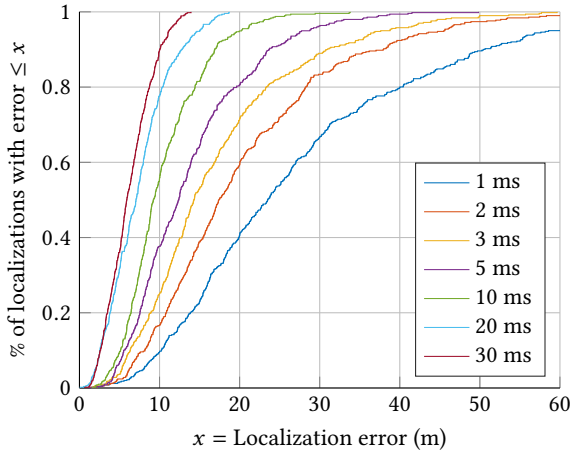


Figure 6: Comparison of positioning accuracy when averaging over different numbers of consecutive fixes.

Time Protocol (NTP). The start time of subsequent one millisecond windows was estimated by counting the number of elapsed samples in the recorded data stream.

To evaluate the accuracy of our algorithm, we placed the receiver antenna on a survey point located on our university building. The location of this point is known accurately. We expect errors in its location to affect our results negatively giving us a lower bound of the performance.

Unless otherwise indicated, experiments were performed under good signal conditions (direct line of sight to most satellites above the horizon) and the search space size was 200 km × 200 km × 30 km × 10 s. The reason for the size of the search in the time dimension is that with a low energy oscillator with maximum drift of 5 ppm and an initial time error of 50 ms (easily achievable with NTP), a range of ±5 seconds covers a duty-cycle interval of more than 11 days. So, ±5 seconds are a large bound on the time search especially since time inaccuracies can be compensated when a fix is computed.[3]

For each processed one millisecond window of signal, we varied the grid of hypotheses uniformly at random in each dimension, up to half the distance between two points. This eliminates possible bias from a specific positioning of the grid. For instance, if one hypothesis always matched the correct receiver position and time exactly, the results might look much better than if the correct position and time lie in the center between the closest hypotheses in each dimension.

5.1 Averaging Over Likely Hypotheses

First, we evaluate how the accuracy depends on the number of most likely points used to compute the weighted average as described in Section 4.7. Figure 5 shows the cumulative distribution functions of 501 fixes covering approximately 500 seconds with the duty cycle of 0.999 s. The shown numbers of points are $\{1, \ldots, 7\}$ to the power of four, since we expect the points around the correct position to have the highest likelihoods. So, the idea is that the curves show the results when averaging over the hypercubes in four dimensions with side lengths of one to seven hypotheses around the correct position.

The best accuracies are achieved with 81 or 256 points. Since lower number of points correspond to a higher likelihood threshold to eliminate regions of hypotheses with low maximum likelihood (see Section 4.8), we use 81 points in the following, as this will save more computation time.

Note that existing CD methods search for the best point only, which is clearly suboptimal. The median position error with 81 points is 23.5 m, which is almost twice as good as the solution with the best point only, which has a median error of 44.3 m. The standard deviation is 17.3 m with 81 points and 27.6 m with the best point only. This shows that our weighted averaging is a substantial improvement over standard CD, substantially improving accuracy.

5.2 Position Averaging over Time

To understand the trade-off between accuracy and the amount of data used, we tested the influence of averaging multiple positions computed from different one millisecond long windows (sliding window average). The results – obtained again from 501 windows – are shown in Figure 6. With just a few more milliseconds, we can gain significant accuracy. For instance, with two milliseconds of data, the median positioning error drops from 23.5 m to 17.4 m. With 10 ms, it even drops to 9.2 m. And with 30 ms, *all* positions are within 13.9 meters.

5.3 Horizontal Positioning

To evaluate the accuracy when searching in space only horizontally, we fixed the altitude for the search to that of the ground truth. This

[3]We think a more reasonable upper bound on the duty-cycle interval would be one day, which means that with such a large time search, we could tolerate many failed localizations between two successful ones, for instance when the receiver is indoor for a long time period.

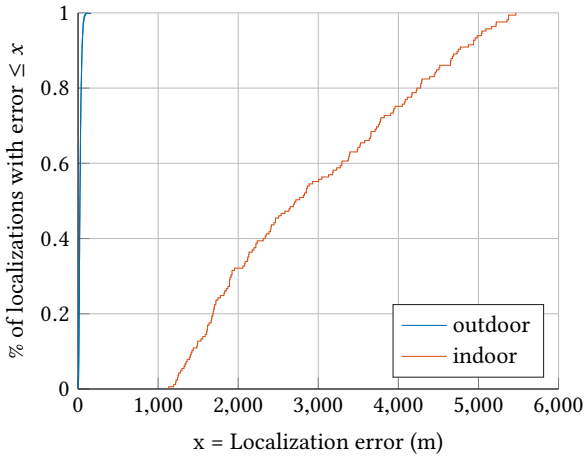


Figure 7: Positioning accuracy with different signal qualities. Outdoors the solution is much more accurate than indoors.

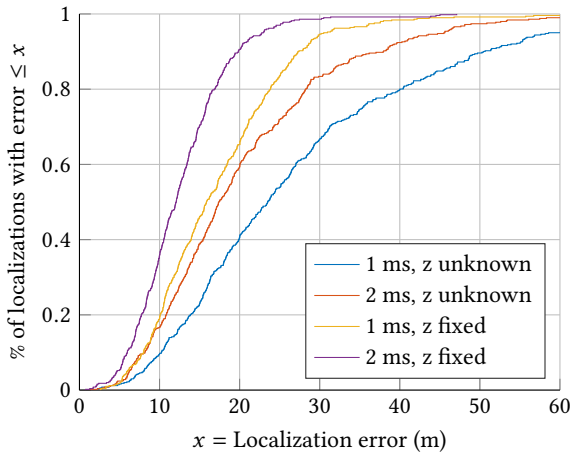


Figure 8: Positioning accuracy with and without fixed height.

emulates scenarios where the receiver is 1) on the Earth surface, so the height can be determined using an Earth elevation model (for example the United States Geological Survey (USGS) elevation model[4]) or 2) the receiver has a barometer, whose measurements can be used together with meteorological data to determine the altitude. The benefit of such an approach is not only better accuracy, as can be seen in Figure 8, but the search space is reduced by one dimension, resulting in less hypotheses to test, which translates to faster and less energy consuming processing. For the positioning with fixed height, we also first determined the best number of points for weighted averaging with the same procedure as explained in Section 5.1, although with numbers to the power of three, because the search space is three dimensional. The best number of points turned out to be 64. Also for this experiment, the number of one milliseconds windows processed was 501.

The idea of using an Earth elevation model to restrict the possible solutions has also been used by Liu et al. [6]. Because we do not have an implementation of CTN available, we cannot directly compare our results to theirs. However, the box plots in their paper show a median error of approximately 40 m with 2 ms of data used. Our median error when using 2 ms of signal and fixing the height of the solution is 12.1 m. This suggests that our approach is competitive.

5.4 Computation Time

To show how the performance of our method using branch-and-bound depends on the signal conditions, we conducted two experiments capturing both very good signal conditions (rooftop) as well as very bad signal conditions (inside a multistory university building). We reduce the search space to 10 km × 10 km × 1 km × 4 s for this experiment, to also be able to test the brute force implementation which tests every single hypothesis.

Figure 7 shows the cumulative distribution functions in both indoor and outdoor scenarios as described in the last paragraph. It clearly shows that the indoor scenario did not allow for a meaningful localization.

For the indoor test, the computation time was 240 s, whereas in very good conditions, the time is only 18.6 s. Note that the indoor test presents a worst case scenario in both computation time and localization accuracy. The brute force implementation takes more than two hours to complete in any scenario.

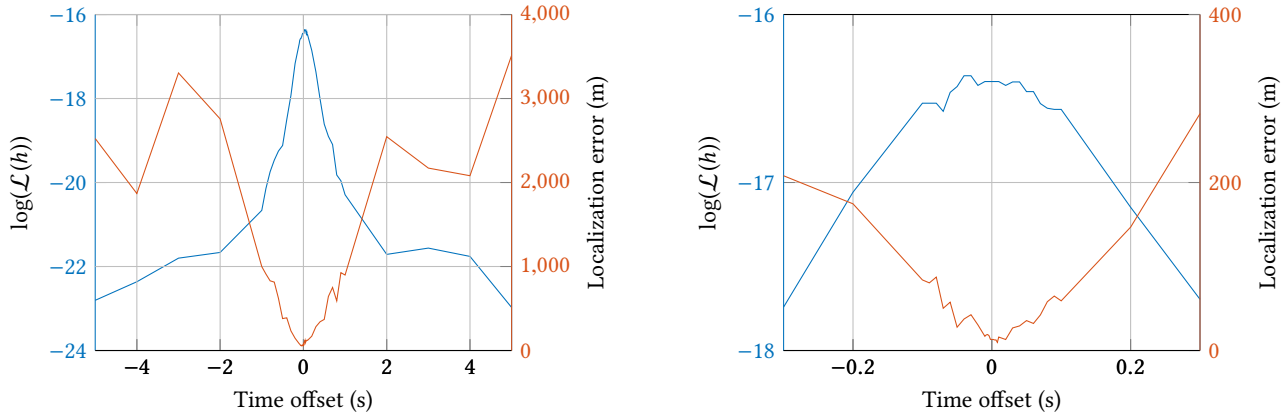
This means that even in situations that make it difficult to find a fix, we find the most likely location in reasonable time compared to a brute force implementation. For the previous experiments with the larger grid in good signal conditions, our method takes 31 s of computation time.

The performance corresponds to the execution on a current Intel i7 mobile processor with a single thread. The runtimes are not indicative of an optimized implementation of our method, since it could easily be parallelized because the computation of the likelihood is independent for each hypothesis. In all the above experiments about computation time, roughly $2 \cdot 10^4$ hypotheses are evaluated each second. A working CUDA implementation of the brute force method revealed that on a Nvidia GTX 1080, roughly $2 \cdot 10^6$ hypotheses can be evaluated each second which indicates that the search can be sped up 100 times. Therefore, an initial fix can be computed in significantly less than a second under good signal conditions. Note that tracking a receiver is cheaper because the search space is smaller.

5.5 Time Dependence of the Likelihood Function

To test the influence of the time parameter in our likelihood function, we picked a random one millisecond long window of the sampled signal and searched the position which maximizes the likelihood given different receiver times. The results are shown in Figure 9, other ms windows exhibit the same properties as described below. The plot to the left shows that, at least for signals with good quality, our likelihood function (in blue) is roughly convex in the

[4]More information about the USGS elevation model can be found at the "The National Map" website: http://nationalmap.gov/elevation.html



(a) Large time errors.

(b) Small time errors.

Figure 9: Shape of $\mathcal{L}(h)$ for different time errors as well as the corresponding localization error. For each time offset, the values of the hypothesis h with maximum $\mathcal{L}(h)$ are indicated.

time dimension. However, we cannot reconstruct the correct time very precisely because the probability of the best point does not change significantly when the time is within a second of the correct time (blue curve in the right hand side plot). However, the positioning quality varies significantly inside this time range (orange curve in the right hand side plot). This is due to the fact that within the search space, there are points which still match the received signal very well. Judging from the localization error, the most likely position passes the correct position in a linear fashion as the time error is varied from negative to positive. This suggests that the likelihood function is quite flat in the time domain which is one of the reasons why the averaging over the most likely hypotheses helps to increase the accuracy of our method. As a side note, this last observation could lead one to think that the correct hypothesis can also be found at the center of the likelihood plateau. However, the localization error scales in Figure 9 are relatively large, meaning that the center would have to be found very accurately. In fact, we experimented with fitting different shapes to the likelihood function, but this yielded results with a large variance.

6 CONCLUSION

We showed how Collective Detection can be optimized to achieve performance that allows for very coarse initial guesses for both position and time. Our branch and bound method scales well in both good as well as bad signal conditions. The localization performance is superior to similar approaches due to the averaging which greatly reduces the effect of the flatness of the likelihood function. When utilizing more than one millisecond of signal, the performance is very competitive even with classical GPS receivers consuming much more energy.

Our method allows for a snapshot GPS receiver design, which only samples one millisecond of signal per fix and can run on a low power 5 ppm oscillator. Therefore, the energy per fix will be extremely low - since the computation can be done in the cloud - and the receiver can also support very small duty cycles, for instance 10ms per fix once every hour. This gives our method an advantage in terms of energy usage over the classical approach, which samples the signal continuously. Note that such a snapshot receiver does also not need a large and heavy radio time signal antenna like the design presented by Liu et al. [6].

ACKNOWLEDGMENTS

We sincerely thank our shepherd Polly Huang, as well as the anonymous reviewers, for their valuable suggestions and feedback.

REFERENCES

[1] Dennis Akos. 1997. *A Software Radio Approach to Global Navigation Satellite System Receiver Design.* Ph.D. Dissertation. Ohio University, Athens, OH.
[2] Penina Axelrad, Ben K Bradley, James Donna, Megan Mitchell, and Shan Mohiuddin. 2011. Collective detection and direct positioning using multiple GNSS satellites. *Navigation* 58, 4 (2011), 305–321.
[3] Joon Wayn Cheong, Jinghui Wu, Andrew G Dempster, and Chris Rizos. 2011. Efficient implementation of collective detection. In *IGNSS symposium.* 15–17.
[4] Pau Closas, Carles Fernández-Prades, and Juan A Fernández-Rubio. 2007. Maximum likelihood estimation of position in GNSS. *Signal Processing Letters, IEEE* 14, 5 (2007), 359–362.
[5] United States of America Department of Defense. 2008. *Global Positioning System Standard Positioning Service Performance Specification (GPS SPS PS), 4th Edition.* Technical Report.
[6] Jie Liu, Bodhi Priyantha, Ted Hart, Heitor S. Ramos, Antonio A.F. Loureiro, and Qiang Wang. 2012. Energy Efficient GPS Sensing with Cloud Offloading. In *Proceedings of the 10th ACM Conference on Embedded Networked Sensor Systems (SenSys 2012).* ACM, 85–98.
[7] Carey E Noll. 2010. The Crustal Dynamics Data Information System: A resource to support scientific analysis using space geodesy. *Advances in Space Research* 45, 12 (2010), 1421–1440.
[8] James Bao-Yen Tsui. 2000. *Fundamentals of Global Positioning System Receivers - A Software Approach.* John Wiley and Sons, Inc., New York, NY.
[9] Frank Stephen Tromp Van Diggelen. 2009. *A-GPS: Assisted GPS, GNSS, and SBAS.* Artech House.