# Appendix D - Fence Controller (Matt Grogan)

The fence controller subsystem is responsible for processing data received by the user interface and the mobile tracking unit. The subsystem is entirely software based and will run on the BeagleBone Black. It consists of three conceptually distinct units: communication with the user interface, processing data received from the mobile tracker and the base station, and integrating the results of the previous units to determine what action, if any, should be taken.

The fence controller will receive requests for the location of the mobile tracker and new boundaries for the virtual fence. Outputs will consist of alerts (low battery on mobile tracker and mobile tracker crossing boundary) and, upon request, the location of the mobile tracker. The processing of location data received by the mobile tracker and the base station will be accomplished using Real Time Kinematic (RTK), a method for precise positioning with GNSS. Errors in the reported position of the mobile tracker will be filtered to an acceptable level and this corrected location will be passed along for further use. Information from a battery monitor on the mobile tracker will also be received. Once the data from the mobile tracker has been parsed and processed, an action will be taken based on these data and the use inputs.

1. Functional Specifications
   1. Functional Requirements
      1. The fence controller subsystem shall determine the location of a mobile tracker with an accuracy of no less than 1 meter while the mobile tracker is stationary. The same accuracy should be expected for a moving target, but will not be promised due to potential difficulties in testing.
      2. The fence controller subsystem shall determine if a mobile tracker is within a boundary, received from a user interface, consisting of no less than 3 and no more than 100 points. Additionally, the two most distant points will be separated by no more than 0.9km. In the event that the mobile tracker is not within the boundary, the user interface and the mobile tracker shall be notified.
      3. The fence controller subsystem shall, upon request from the user interface, report the most recent known location of the mobile tracker.
      4. The fence controller subsystem shall notify the user interface in the event that the battery charge, as reported by the mobile tracker, falls below a 10% threshold.
   2. Constraints and Assumptions
      1. In order to precisely determine position, the RTK technique requires a base station with a well-defined location to provide real-time corrections. As such, error in reported base station location will propagate and affect the calculated position of the mobile tracker. The positioning algorithm will assume an accurate base station position.
      2. Positioning will depend on current data from the mobile tracker. Thus, a stable communication channel is assumed.
2. Operation

The fence controller subsystem consists of software for real-time processing of input from the base station, mobile tracker, and user interface. The platform on which this software will run is the BeagleBone Black which incorporates a 1GHz ARM Cortex-A8 processor. Raw GPS data from both the base station and mobile tracker are used to calculate a precise location with RTK methods. This location will be checked against a boundary instantiated by a user. Additional inputs are the battery status of the mobile tracker and user inputs for modifying the boundary and checking location. Outputs are the user requested location and notifications in the event of a boundary breach or low battery. A block diagram depicting the subsystem is shown in Figure D-1.



Figure D-. Subsystem block diagram

1. RTK Positioning
2. GNSS Signal Measuring

Since RTK positioning depends on carrier wave phase rather than signal content, this is worth briefly exploring. GNSS signals are composed of a multiplication of carrier frequency, spreading code, and navigation data.

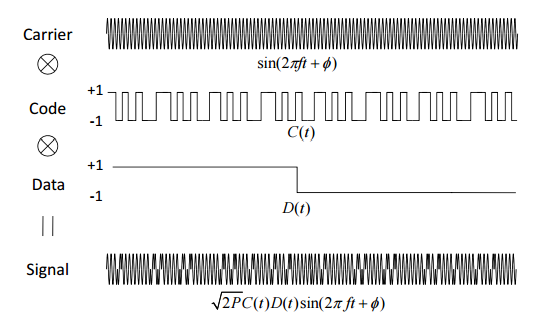


Figure D-. GNSS signal structure

Pseudorange is defined as the distance from a receiver antenna to a satellite antenna including the error due to receiver/satellite clock offsets and atmospheric delays. The pseudorange for satellite can be expressed in terms of the signal reception time as measured by the receiver clock and the signal transmission time as measured by the satellite clock :

) (1)

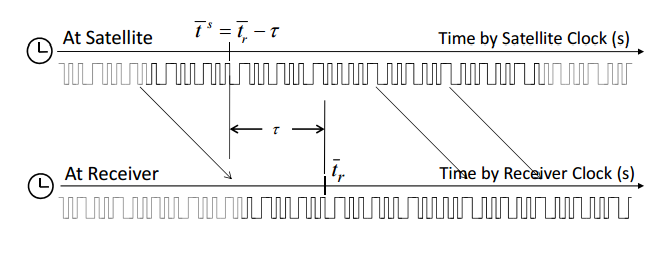


Figure D-. Pseudorange model

This can be rewritten in terms of geometric range between satellite/receiver antennas, satellite/receiver clock offsets and , ionospheric/tropospheric delays and , and the measurement error :

(2)

Carrier phase is a measure of the distance from a satellite antenna to a receiver antenna expressed in terms of the number of cycles of the carrier frequency. Since the carrier wavelength is small (~20cm) this measure is precise within the carrier wavelength but inaccurate due to an integer ambiguity in the number of cycles. The carrier phase can be written in terms of a receiver oscillator phase , navigation signal phase , carrier phase integer ambiguity , and measurement error :

(3)

The phase range is the carrier phase multiplied by the carrier wavelength :

(4)

1. Positioning Algorithms

The previous section contained equations that are generally true for all RTK systems. The following section will contain equations more applicable to our design, taking into account factors such as single frequency GPS units, short baseline length, and our use of the RTKLIB library. In RTKLIB, all positions are internally represented as x, y, and z components in an ECEF (Earth-centered, Earth-fixed) coordinate system, taking the point (0,0,0) to be the center of the Earths mass. For the sake of clarity, the mobile tracker will be referred to in this section as a rover – this is the conventional nomenclature.

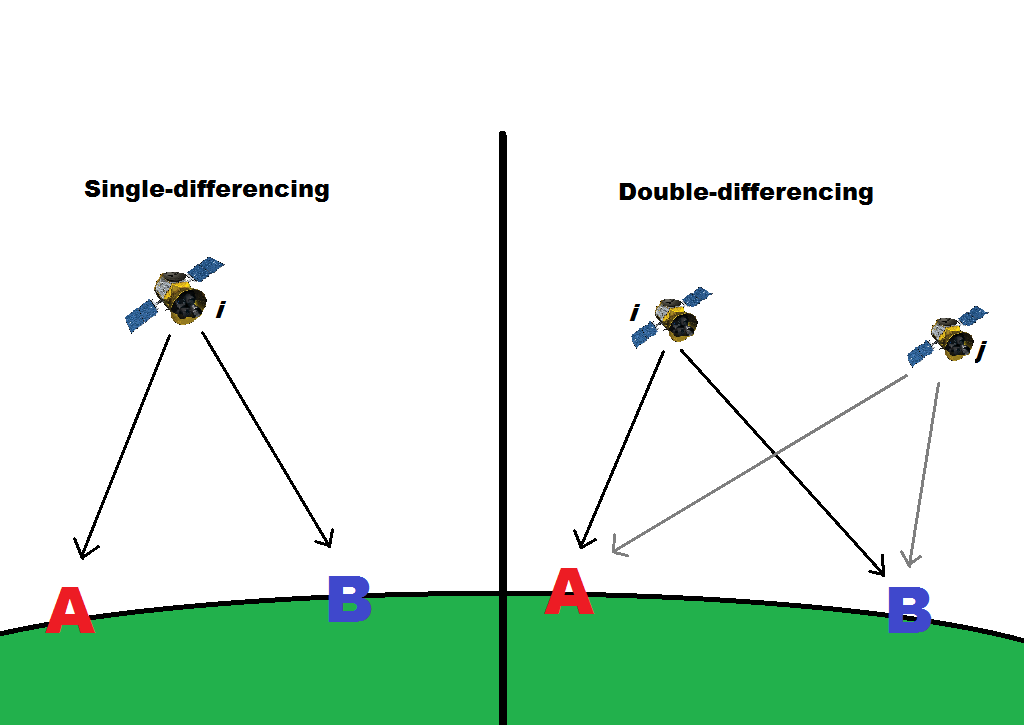


Figure D-. Single and double differencing

When a rover *r* and base station *b* pair have a short baseline separation, typically less than 10km, a double-differencing technique will eliminate atmospheric and clock biases. In that case, the phase range and pseudorange from equations (2) and (3) may be expressed as:

(5)

(6)

where represents the single-difference between satellites, represents the single-difference between receivers, and is the single difference of carrier phase ambiguities within a cycle. Important for the RTKLIB implementation is the extended Kalman Filter (EKF), a nonlinear extension of the Kalman Filter. The Kalman Filter is a recursive algorithm for generating a statistically optimal estimate of a system state using previous (and possibly inaccurate) input, a prediction model, and an observation model. We use the EKF due to the nonlinear nature of real datasets. In our positioning model, we have an unknown state vector defined as:

(7)

where is the rover antenna position in the ECEF frame. A measurement vector taken at epoch, or reference time, is defined with double-differenced phase range and pseudorange as:

(8)

Now, the state vector and its covariance matrix can be estimated using the EKF. Note that indicates an estimated state and indicate before and after the EKF measurement update respectively:

(9)

(10)

(11)

where is the measurements model vector, is the matrix of partial derivatives, is the covariance matrix of measurement errors, and is a gain term which weights the values of the previous estimate and the current observation. The definitions of the above vectors and matrices are as follows:

(12)

,

,

(13)

(14)

where is the position of satellite *i* in the ECEF frame, is the position of the base station in the ECEF frame, is the line of sight vector to satellite *i* from the rover, is the single-differencing matrix, and σ is the standard deviation of carrier phase/psuedorange error. At each new epoch , the new “previous” state estimate and covariance matrix must be updated as follows:

(15)

(16)

,

where is the state transition matrix and is the covariance matrix of system noise, in this case modelling white noise. Solving equations (9-11) for the state vector yields the estimated rover antenna position and single-differenced carrier phase ambiguities. However, at this point the estimated state will be in floating point form. Since carrier-phases are periodic, a float solution is not ideal and will result in reduced accuracy. To overcome this, we convert to a double-differenced form:

(17)

(18)

(19)

where now indicates the double-differenced form, is the double-differenced carrier phase ambiguity, and is the double-differencing transformation matrix. By double-differencing, receiver initial phase terms are cancelled and is integer valued. In RTKLIB, the optimal integer vector which satisfies the integer least square problem is determined using the Least-squares Ambiguity Decorrelation Adjustment (LAMBDA) algorithm. Finally, the best estimate for the rover position is determined with:

(20)

1. Boundary Monitoring.

The task of determining a boundary breach by the mobile tracker presents itself as a point-in-polygon (PIP) problem. The possibilities for the mobile tracker are that it is: inside the boundary, on the boundary, or outside the boundary. Only the third case will be considered a boundary breach. To solve the problem, the ray casting algorithm will be used. A ray beginning at the location of the mobile tracker is extended a sufficient distance in any direction. If this ray intersects the polygon edge an odd number of times, it must be within the boundary. A flowchart depicting program execution is shown in Figure D-5.



Figure D-. Boundary monitor flowchart

The above loop will be executed once per second in order to keep the state of the mobile tracker current. Due to the cheapness of ray casting, CPU usage for execution at 1Hz will be negligible. This is an important consideration given that the processor will be simultaneously running a graphical user interface and the computationally intensive RTK algorithms.

1. Battery Monitoring

The battery monitor data will consist of 16 bit messages encoding a voltage level. Should the voltage level drop below 10% of capacity, the user interface will receive a notification.

1. I/O
2. User Interface

Due to very recent changes in the location and implementation of the user interface, our current strategy for inter-subsystem communication is in flux. Going forward, the most likely solution will incorporate TCP/IP. This field will be updated ASAP.

1. Base Station

The fence controller subsystem will interface with hardware located on the base station through device drivers. These are within the domain of the base station subsystem. I/O should be as simple as reading a device file from within a program.