ASE 372N Laboratory #1 Report
In-field data collection with a handheld GPS receiver

Caleb North

September 8, 2016

1 Abstract

An error analysis was performed on NMEA 0183 position solutions recorded *in situ* using a Trimble Juno SB handheld GPS receiver. To determine how the WAAS functionality affects performance, the precision and accuracy of the receiver with and without the WAAS capability enabled was assessed. It was found that the inclusion of WAAS did not significantly alter the accuracy, however the horizontal circular error probable (CEP) was reduced by 25%. To observe how environment impacts performance, data was collected at two separate sites with different horizontal profiles. It was found that the site with less obscura reduced the CEP by 34%. Furthermore, it was found that measurement precision was sensitive to satellite geometry. It was shown that a data-set with more satellites in view could reduce the CEP by 38%.

2 Introduction

This lab observed how precision and accuracy can be affected by various factors: WAAS functionality enabled/disabled, horizon profile, and GPS constellation geometry. To accomplish this, a Trimble Juno SB handheld GPS receiver, with the SirfTech application installed, was used to record NMEA 0183 position estimates. A total of six 10-minute data collections were carried out at 3 different locations around the University of Texas at Austin. Table 1 shows when and where (local time) these data collections occurred.

Test Name	Location	Start Time
WAAS Enabled	South Mall	6:04pm
WAAS Disabled	South Mall	6:06pm
Site 83: Test 1	24th St. and San Jacinto Ave.	8:09pm
Site 83: Test 2	24th St. and San Jacinto Ave.	9:52pm
Site 56: Test 1	South-West corner of ECT	9:21pm
Site 56: Test 2	South-West corner of ECT	10:18pm

Table 1: Data-set names, locations, and start times.

For the data processing, a Matlab function `readNMEAposV3.m` was used to convert the raw NMEA message logs to csv files containing: seconds of day, latitude, longitude, altitude, and number of satellites tracked. Additionally, three Python scripts were written: 1) `lla2ecef.py` converts a position in the LLA frame (Latitude-Longitude-Altitude) to the ECEF frame (Earth Centered Earth Fixed), 2) `ecef2enu.py` calculates the rotation matrix to convert from the Cartesian ECEF frame to the geodetic ENU frame (East-North-Up), and 3) `Lab1.py` is an analysis script tailored for this study, which calculates and plots the means, standard deviations, CEP (circular error probable) for the 2D horizontal, and SEP (spherical error probable) in each frame for each data-set. These scripts can be found in Appendix A.

3 Results and Analysis

When the `readNMEAposV3.m` function was fed the test log files, it produced LLA positions. Table 2 shows the mean values for those positions and SV count.

Mean Values	Latitude (rad)	Longitude (rad)	Altitude (m)	Satellites Tracked
WAAS Enabled	0.528578693	-1.70587545	178.8	9.0
WAAS Disabled	0.528578740	-1.70587534	178.8	9.3
Site 83: Test 1	0.528608035	-1.70577042	159.4	8.7
Site 83: Test 2	0.528608272	-1.70577039	152.8	7.8
Site 56: Test 1	0.528655580	-1.70580959	154.1	7.9
Site 56: Test 2	0.528653904	-1.70581267	204.1	6.6

Table 2: Mean position values in the LLA frame and mean number of satellites tracked.

The script `lla2ecef.py` was used to convert each of the LLA position measurements to ECEF positions. The average ECEF position for each data-set was computed and is shown in Table 3

Mean Values	X (Km)	Y (Km)	Z (Km)	Satellites Tracked
WAAS Enabled	-742.365067	-5462.31250	3197.81682	9.0
WAAS Disabled	-742.364432	-5462.31243	3197.81707	9.3
Site 83: Test 1	-741.776405	-5462.28071	3197.96797	8.7
Site 83: Test 2	-741.775362	-5462.27429	3197.96592	7.8
Site 56: Test 1	-741.969267	-5462.09619	3198.22607	7.9
Site 56: Test 2	-741.992631	-5462.14200	3198.24210	6.6

Table 3: Mean position values in the ECEF frame and mean number of satellites tracked.

Removing the average from the ECEF positions resulted in a collection of ECEF residuals. The residuals where then converted to ENU and the standard error metrics where calculated (See Appendix A). Table 4 shows error metrics for position and number of tracked satellites for each data set.

	σ_{East} (m)	σ_{North} (m)	σ_{Up} (m)	CEP (m)	SEP (m)	Satellites Tracked
WAAS Enabled	0.48	0.61	1.82	0.65	1.52	9.0
WAAS Disabled	0.70	0.75	2.19	0.86	1.86	9.3
Site 83: Test 1	1.54	0.95	3.10	1.50	2.76	8.7
Site 83: Test 2	2.12	2.37	6.18	2.65	5.34	7.8
Site 56: Test 1	1.35	2.36	3.63	2.27	3.49	7.9
Site 56: Test 2	2.43	1.01	7.44	2.20	6.07	6.6

Table 4: Error metrics for each data-set.

By inspection, it is observed that data-sets which were tracking more satellites appear to have less statistical error, i.e. more precision.

3.1 WAAS

The WAAS satellite provides differential corrections to the GPS satellites. So, it is expected that the use of the WAAS functionality would provide a better solution, i.e. more accuracy and precision. This section will determine how much of an improvement was observed in the measurements. Figure 1 shows the ENU residuals for the receiver with and without the WAAS option enabled.

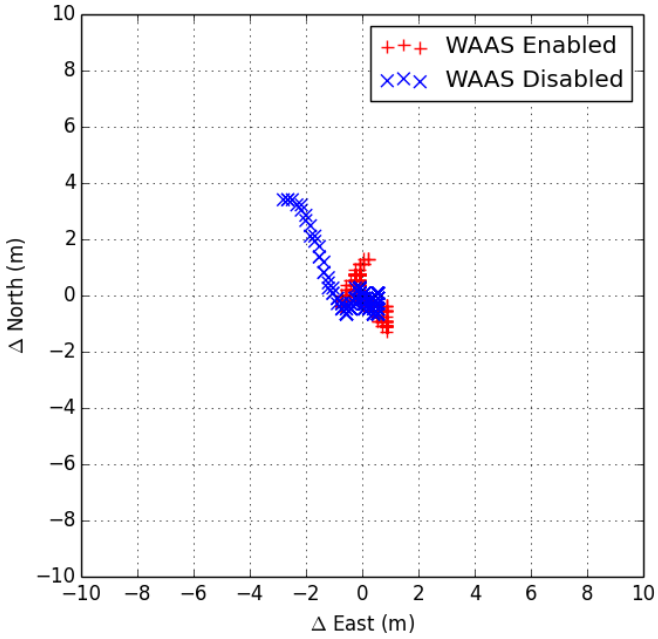

Figure 1: ENU residuals with and without WAAS enabled.

Table 5 displays the error metrics, comparing the WAAS enable/disable data-sets. From inspection, it is observed that enabling the WAAS functionality reduced the CEP by 25%.

	σ_{East} (m)	σ_{North} (m)	σ_{Up} (m)	CEP (m)	SEP (m)
WAAS Enabled	0.48	0.61	1.82	0.65	1.52
WAAS Disabled	0.70	0.75	2.19	0.86	1.86

Table 5: Error metrics with and without WAAS enabled.

3.2 Satellite Geometry

As time progresses the GPS satellites sweep out tracks in the sky. This means that the geometry of the GPS constellation is ever-changing. A consequence of this phenomenon, is that the precision GPS position measurements is also changing, especially in urban canyons which can be found throughout the UT campus. Two data collects where taken nearly 2hrs apart. The analysis presented in this section is intended to determine how the precision is affected.

Figure 2 shows the position residuals, with respect to the mean, on the position estimates in the LLA frame on the left, and ENU frame on the right. A cursory comparison shows that the two frames have similar behavior, which is to be expected.

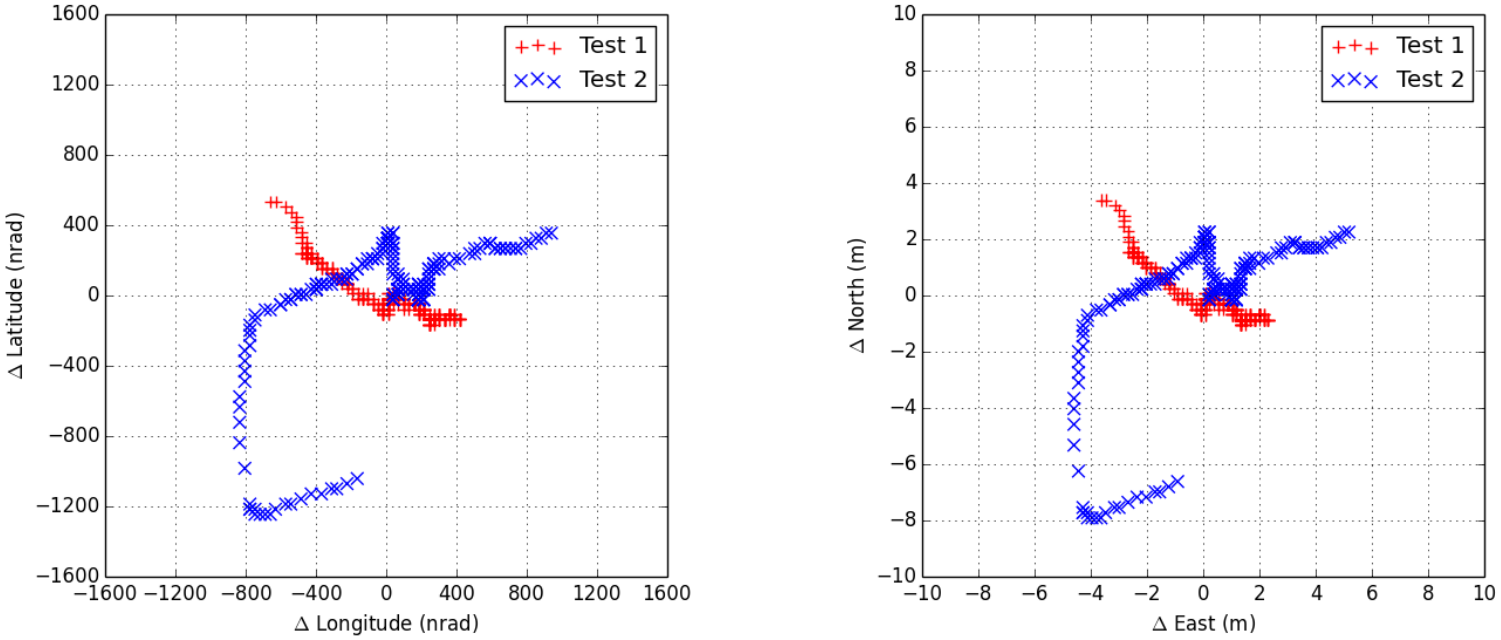

Figure 2: Residuals from the mean for Test 1 and 2 in the LLA frame (left) and ENU frame (right).

Table 6 displays the error metrics, comparing data-sets collected from the same location, but 2 hours apart. From inspection, the site with less obscura reduced the CEP by 34%. The precision of the receiver can be interpreted to be

When focusing on Site 83: Test 1, we find the standard deviation of the vertical height residuals to be on the order of 3m, while the CEP of the horizontal is 1.5m. This indicates that the vertical

	σ_{East} (m)	σ_{North} (m)	σ_{Up} (m)	CEP (m)	SEP (m)	Rx's Error (m)
Site 83: Test 1	1.54	0.95	3.10	1.50	2.76	3.5
Site 83: Test 2	2.12	2.37	6.18	2.65	5.34	6.8

Table 6: Table illustrates...

error is twice as much as the horizontal, which can be understood with the realization that the vertical component only has satellites in the zenith. Another way of stating this is that the VDOP is larger than the HDOP. The SEP is reported to be 2.76m, while the receiver's internal error estimate was recorded to be 3.5m. This could be understood if the internal estimator is using the 3D-RMS which can be approximated as: 3D-RMS = 1.3×SEP = 3.6m. This would give very good agreement with the computed value.

Google Earth was used to overlay the position measurements (LLA) with imagery. Site 83's mark is indicated in the figure by a red circle. The accuracy of Google Earth's rendering appears to be on the order of 3m.

Figure 3: Google Earth generated imagery showing Site 83 (red) with Test 1 data overlaid (yellow).

3.3 Horizon Profile

Sky visibility is is crucial for achieving high precision with GPS receivers. This section will observe how the measurement precision behaves, when measurements are taken at two different locations, each with unique horizon profiles. Table 7 shows a comparison of two site with different sky-lines. From inspection, Site 83 has a 38% reduction in CEP when compared to Site 56. It is also apparent that the site with more precision is tracking 1 more SV for the majority of the data-set.

	σ_{East} (m)	σ_{North} (m)	σ_{Up} (m)	CEP (m)	SEP (m)	Satellites Tracked
Site 83: Test 1	1.54	0.95	3.10	1.50	2.76	8.7
Site 56: Test 1	1.35	2.36	3.63	2.27	3.49	7.9

Table 7: Table illustrates...

Figure 4 shows how Google Maps was used to estimate a distance of 370m between Site 83 and Site 56. While, the magnitude of the difference in ECEF position between 'Site 83: Test 1' and Site 56: Test 1' indicates that the separation distance is 371m.

Figure 4: .

4 Conclusion

A quantitative analysis was performed on NMEA 0183 position estimates taken from a Trimble Juno SB handheld GPS receiver. To determine how the WAAS functionality affects performance, the precision and accuracy of the receiver with and without the WAAS capability enabled was assessed. It was found that the inclusion of WAAS did not significantly alter the accuracy, however the circular error probable (CEP) for the horizon was reduced by 25%. Data was collected at two separate sites with different horizontal profiles to observe how environment impacts performance. It was found that the site with more sky-coverage reduced the CEP by 34%. Furthermore, it was found that precision was sensitive to satellite geometry. It was shown that a site could reduce the CEP by 38% by just have 1 more satellite in view.

References

[1] P. Misra and P. Enge, *Global Positioning System: Signals, Measurements, and Performance.* Lincoln, Massachusetts: Ganga-Jumana Press, revised second ed., 2012.

[2] Noureldin, A; Karamat, T.B.; Gregory, J., *Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration* 2013, XVIII, 314p. Hardcover

5 Appendix A

Scripts will be attached individually.

```
%  Original by: Robert B Harris
%  Modified for Matlab by: Hector Escobar
%  Last Modified: September 18, 2010
%  Modified: September 16, 2010.
%  Modifications: Addapted for Matlab. Fixed bug where sod didn't show up.
%  Number of satellites tracked added.

%  [sod, lats, longs, hts, sats] = readNMEApos(fname)
%
%     Reads NMEA messages recorded to a text file, and transforms the GPGGA
%     (position fix) messages into four vectors:
%
%        sod   -- Seconds of day.
%        lats  -- Latitude, with decimal fractions, in whole degrees.
%                 North is positive.
%        longs -- Longitude, same units as latitude. East is positive.
%        hts   -- Height above ellipsoid, in meters.
%        sats  -- Number of satellites in view

function [sod, lats, longs, hts, sats] = readNMEAposV3(fname)
fid = fopen(fname,'rt');

numRead = 0;
numGood = 0;
numGoodPos = 0;

% First read -- count number of spaces

tline=fgetl(fid);
while ischar(tline)

   numRead = numRead+1;

   % Extract the recorded checksum
   csumidx = findstr(tline,'*');
   ocsum = hex2dec(tline(end-1:end));

   % Comput expected checksum
   ccsum = 0;
   for k=2:(csumidx-1)
     ccsum = bitxor(ccsum,abs(tline(k)));
   end;

   if (ocsum==ccsum)
      numGood = numGood+1;
      if (tline(2:6)=='GPGGA')
         numGoodPos = numGoodPos+1;
      end
   end

   tline = fgetl(fid);
end

fclose(fid);

fprintf('Read %d lines of data\n', numRead);
fprintf('%d passed checksum.\n', numGood);
fprintf('%d good position fixes.\n', numGoodPos);


% Second read -- store GPGGA information
sod = zeros(numGoodPos,1);
lats = sod;
longs = sod;
hts = sod;
sats=sod;
fid = fopen(fname);
tline=fgetl(fid);

idx = 0;

while ischar(tline)
```

```
   % Extract the recorded checksum
   csumidx = findstr(tline,'*');
   ocsum = hex2dec(tline(end-1:end));

   % Comput expected checksum
   ccsum = 0;
   for k=2:(csumidx-1)
     ccsum = bitxor(ccsum,abs(tline(k)));
   end;

   if (ocsum==ccsum)
      if (tline(2:6)=='GPGGA')
         idx = idx+1;
         cidx = findstr(tline,',');
         sodStr = tline((cidx(1)+1):(cidx(2)-1));
         latStr = tline((cidx(2)+1):(cidx(3)-1));
         latDirStr = tline(cidx(3)+1);
         lonStr = tline((cidx(4)+1):(cidx(5)-1));
         lonDirStr = tline(cidx(5)+1);
         htStr = tline((cidx(9)+1):(cidx(10)-1));
         sats(idx)=str2num(tline((cidx(7)+1):(cidx(8)-1)));
         sod(idx) = str2num(sodStr(1:2))*3600 + str2num(sodStr(3:4))*60+str2num(sodStr(5:
(length(sodStr)-4)));

         latSgn=1;
         if (latDirStr=='S')
            latSgn=-1;
         end

         lonSgn=1;
         if (lonDirStr=='W')
            lonSgn=-1;
         end


         lats(idx) = latSgn*(str2num(latStr(1:2))+str2num(latStr(3:length(latStr)))/60);
         longs(idx) = lonSgn*(str2num(lonStr(1:3))+str2num(lonStr(4:length(lonStr)))/60);
         hts(idx) = str2num(htStr);
      end
   end

   tline = fgetl(fid);
end

fclose(fid);
```

```
#!/usr/bin/env python

from numpy import *

def ecef2enu(lat, lon):
#+============================================================================+
# ecef2enu : Generate the rotation matrix used to express a vector written in
#             ECEF coordinates as a vector written in local east, north, up
#              (ENU) coordinates at the position defined by geodetic latitude
#             and longitude.
# INPUTS
#
# lat (phi)   ----- geodetic latitude in radians
#
# lon (lamda) ----- longitude in radians
#
#
# OUTPUTS
#
# R ------- 3-by-3 rotation matrix that maps a vector v_ecef expressed in the
# ECEF reference frame to a vector v_enu expressed in the local
# east, north, up (vertical) reference frame as follows:
# v_enu = R*v_ecef.
#
#+----------------------------------------------------------------------------+
# References:
#    1:  Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration
#        Noureldin, A; Karamat, T.B.; Gregory, J.
#        2013, XVIII, 314p. Hardcover
#        ISBN: 978-3-642-30465-1
#        Site: http://www.springer.com/978-3-642-30465-1 ### Needs clarification ##
#
# Author: Caleb North
#+============================================================================+


    R = array([
        [         -sin(lon),           cos(lon),        0.],
        [-sin(lat)*cos(lon), -sin(lat)*sin(lon), cos(lat)],
        [ cos(lat)*cos(lon),  cos(lat)*sin(lon), sin(lat)]
        ])



    return (R)
```

```
#!/usr/bin/env python

from numpy import *

def lla2ecef(lat,lon,alt):
#+============================================================================+
# lla2ecef :     Convert from latitude, longitude, and altitude (geodetic with
#                 respect to the WGS-84 ellipsoid) to a position vector in the
#                 Earth-centered, Earth-fixed (ECEF) reference frame.
#
#
# INPUTS
#
# lat ----- latitude in radians
#
# lon ----- longitude in radians
#
# alt ----- altitude (height) in meters above the ellipsoid
#
#
# OUTPUTS
#
# pVec ---- 3-by-1 position coordinate vector in the ECEF reference frame,
# in meters.
#
#+----------------------------------------------------------------------------+
# References:
#    1:  Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration
#        Noureldin, A; Karamat, T.B.; Gregory, J.
#        2013, XVIII, 314p. Hardcover
#        ISBN: 978-3-642-30465-1
#        Site: http://www.springer.com/978-3-642-30465-1 ### Needs clarification ##
#
# Author: Caleb North
#+============================================================================+

    # WGS84 Values #
    a = 6378137.0            ## Semimajor axis (equatorial radius)
    f = 1/298.257223563        ## Flattening
    b = a*(1-f)                ## Semiminor axis
    e = sqrt(f*(2-f))        ## Eccentricity
    E = sqrt(a**2/b**2 -1)    ##

    N = a/sqrt(1 - e**2*sin(lat)**2)        ## Normal radius

    pVec = array([
        [(N+alt)*cos(lat)*cos(lon)],
        [(N+alt)*cos(lat)*sin(lon)],
        [(N - N*e**2 + alt)*sin(lat)]
        ])


    return (pVec)
```

```
#!/usr/bin/env python

from numpy import *

def ecef2lla(pVec):
#+============================================================================+
# ecef2lla : Convert from a position vector in the Earth-centered, Earth-fixed
#             (ECEF) reference frame to latitude, longitude, and altitude
#             (geodetic with respect to the WGS-84 ellipsoid).
#
#
# INPUTS
#
# pVec ---- 3-by-1 position coordinate vector in the ECEF reference frame,
# in meters.
#
# OUTPUTS
#
# lat ----- latitude in radians
#
# lon ----- longitude in radians
#
# alt ----- altitude (height) in meters above the ellipsoid
#
#    1:  Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration
#
# Caveats: Only good for abs(lat) > .0001 ## more clarification
#
#
#
#+----------------------------------------------------------------------------+
# References:
#    1:  Fundamentals of inertial Navigation, Satellite-based Positioning and their Integration
#        Noureldin, A; Karamat, T.B.; Gregory, J.
#        2013, XVIII, 314p. Hardcover
#        ISBN: 978-3-642-30465-1
#        Site: http://www.springer.com/978-3-642-30465-1 ### Needs clarification ##
#
# Author: Caleb North
#+============================================================================+

    x = pVec[0,0]
    y = pVec[1,0]
    z = pVec[2,0]


    # WGS84 Values #
    a = 6378137.0            ## Semimajor axis (equatorial radius)
    f = 1/298.257223563        ## Flattening
    b = a*(1-f)                ## Semiminor axis
    e = sqrt(f*(2-f))        ## Eccentricity
    E = sqrt(a**2/b**2 -1)    ##

    p = sqrt(x**2 + y**2)
    theta = arctan(z*a/(p*b))

    # Using closed form solution #
    lat = arctan((z+E**2*b*sin(theta)**3)/(p-e**2*a*cos(theta)**3))
    lon = arctan2(y,x)
    N = a/sqrt(1-e**2*sin(lat)**2)    ## Normal Radius
    alt = p/cos(lat) - N

    ## accounting for numerical instabilities ##
    if p  < 1 and z > 0: alt =  z - b
    if p  < 1 and z < 0: alt = -z - b
    if y == 0 and x > 0: lon = 0
    if y == 0 and x < 0: lon = pi
    if p == 0 and z > 0: lat =  pi/2; lon = 0; alt =  z-b
    if p == 0 and z < 0: lat = -pi/2; lon = 0; alt = -z-b


    return (lat, lon, alt)
```