TULIP Algorithm Trilateration with CBG

TULIP results

Faisal generated TULIP results for SLAC as target (134.79.18.188) and TULIP geo-located it to be in Wyoming. Screenshot here. This is obviously way off.

CBG results

Dr. Les explained what TULIP was doing and wanted to do a quick analysis of the same location (i.e. SLAC) using CBG with trilateration. So I found a target located at SLAC (134.79.18.134) in CBG list. We ran CBG with trilateration for this target. The results were way off. The error distance was of the order of ~3200 km.

However Dr. Les pointed out that we should be using three landmarks that have the minimum RTT values from the target. Thus we shuffled our values accordingly and re-ran the test with two different set of landmarks for the same target (SLAC). This helped to verify our results. Table below shows the results:

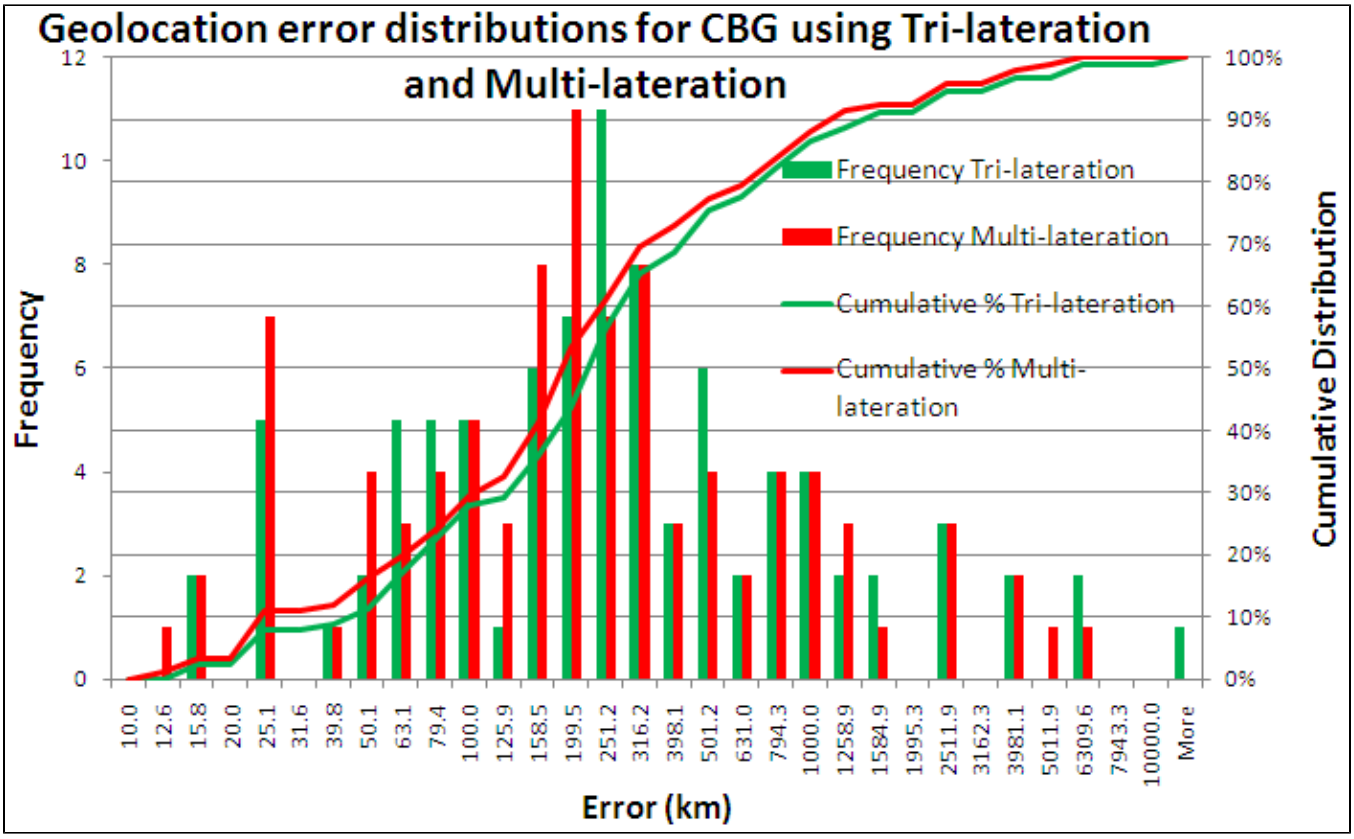
Landmark 1	Landmark 2	Landmark 3	Error (km)	Distance to nearest landmark (km)	Area of Region (km)	Est. Lat /Long	Actual Lat /Long
36.9899 -122.06 (Santa Cruz)	38.4829 -121.64 (Davis)	37.3558 -121.954 (Santa Clara)	25.254	2.5132	1498.5	37.474 -121.93	37.418 -122.2
37.4285 -122.178 (Stanford)	37.3762 -122.183 (Palo Alto)	37.3558 -121.954 (Santa Clara)	2.5156	2.5132	1048.8	37.429 -122.18	37.418 -122.2

CBG with trilateration is performing well.

CBG multilateration vs CBG trilateration comparison

Spreadsheet shows a comparison of error (in km) between CBG multilateration and CBG trilateration. The technique I've followed:

1. Sorting target lists in ascending order on the basis of RTT between the target location and landmark location.
2. Re-running the CBG trilateration code for new results.
3. Spreadsheet and graph for analysis.



It can be seen that both distributions perform similarly. More analysis can be viewed here.

What I didn't do and why:

1. Avoiding duplicate landmarks.

    a. Reason: If you look at the spreadsheet you will notice that there are duplicate entries for multilateration as well. You can infer this from matching Estimated Lat/Longs to Actual Lat/Longs and by observing the distance to the nearest landmark values. Also a few targets don't have more than two landmarks and in all such cases those are duplicates (in terms of Lat/Longs). So in such a case I don't have an option but to use the duplicate ones.
2. Avoiding landmarks present within a target's vicinity.
    a. Reason: Closely related to the point mentioned above.

Results, observations and explanation

In the spreadsheet we have made various calculations in order to understand the results. The following are a few observations and their explanation.

1. Amount of NaNs (in error distance) for multilateration and trilateration

The table below shows number of NaN occurrences for both the techniques.

technique	Number of NaNs (Total targets)
CBG Multilateration	67 (171)
CBG trilateration	11 (171)

What is NaN?

NaN (Not a Number) is a value of numeric data-type representing an undefined or unrepresentable value, especially in floating point calculations. More here .

NaN w.r.t. CBG

According to the CBG code NaNs represent "bad pairs that lead to no region". This means that landmarks that fail to produce intersection regions, consequently also fail to produce an estimate for the location of the target and instead give out an erroneous value. Author has handled such values with NaN (code snippet below).

geolocate.m

```
if constraintType
  warning('Trying speed of light')
  constraintType = 0;
  % switch the constraint type and try again
  [locest,actual,error,regarea,distNearestLandmark,target_id,constraintType,inRegion\] = geolocate(file,
extension,hullbool, constraintType, bestlineTable);
  return;
else
  % find the badPairs that lead to no region, write them to stdout
  badPairs = analyzeNoRegion( measurements )
  %error(\['No SOL intersection region for ', char(file)\])
  region = \[NaN NaN\];
  locest = \[NaN NaN\];
  error = NaN;
  regarea = NaN;
  results = \[target_id error; distNearestLandmark regarea; locest; actual; region\];
  dlmwrite(\[char(file),char(extension)\],results,' ');
  return;
end;
```

I've managed to resolve most of the NaNs by taking only the best possible landmark estimates for each target and/or by eliminating the "bad pairs". Bullets below explain the activity:

- There were a total of 67 NaN values for multilateration.
- Now only 14 NaN values remain for multilateration: 11 are common with trilateration and 3 are unique.
- 39 NaNs removed by keeping number of estimates, n = 10.
- 14 NaNs removed by keeping number of estimates, n = 4.

A sample of "bad pairs" from target "132.248.120.214" is below. Each line contains four values separated by white-space.

First line contains: **target-lat target-long id rtt**. All other lines contain: **landmark-lat landmark-long rtt id**.

Target-132.248.120.214.txt

```
19.2891 -99.1606 86 0
42.3424 -71.0878 0.5 167
47.1544 -88.6471 0.5 119
42.6442 -73.2463 0.5 165
40.4249 -86.9162 0.5 150
28.0631 -82.4128 0.5 142
38.0287 -84.5075 0.5 145
40.7855 -111.737 0.5 166
```

See latest spreadsheet for details.

2. Some results have enormous errors (|error|>1000)

This is true for both multilateration and trilateration. And reasons could be one or more of the following:

- Number of landmarks aren't enough i.e. only 3 or 2 or less.
- Number of landmarks aren't enough and those which are present are duplicates.
- There are enough number of landmarks but none are good enough i.e. the RTT is in the order of 50+ ms (true for error distances in the order of thousands of km).
- There are enough number of landmarks but most of these aren't good enough i.e. the RTT is in the order of 25+ ms (true for error distances in the order of 1000+ km).

These have been inferred from looking at the Target files.

3. Trilateration is performing better than multilateration in some cases

There are 23 instances where trilateration performs better than multilateration, 29 instances where multilateration performs better than trilateration and in the rest both perform equally well. The reason as far as I understand is:

We sorted the Target files on the basis of RTT between the target and landmarks. This promoted those lat/long values to the top of the list which had least RTT from the target. Though any such sorting technique on these values doesn't affect multilateration results but it makes a huge impact on trilateration results. The reason being the way these two techniques use these values. Multilateration considers all values and figures out regions of intersection whereas trilateration simply takes three values to find an intersection region.

So If we have say 10 landmarks and 4 of them had relatively lower RTT to the target, multilateration will give good results. Even if some values aren't really good, it won't cause multilateration to behave in an entirely different way. However in case of trilateration, better the landmark estimates we have, the better the results are. Since trilateration considers three values, even a single one of those three values can make a big difference.

CBG trilateration vs Improved trilateration comparison

Spreadsheet here shows comparison between CBG's trilateration and Farrah's improved trilateration. A few important points:

- There are a total of 174 targets for CBG out of which 131 remain after ignoring values that either have error in the range "error<1km" (i.e. the target and at least one landmark are probably in the same location) or contain "NaN".
- Improved trilateration by Farrah produced results for only 78 targets so far. Her method produced 7 NaN values which she ignored.
- Only 74 targets overlap between CBG trilateration and improved trilateration.
- If I don't ignore CBG's values that have estimate error "error<1km" then CBG trilateration performs 64/74 times better and improved trilateration performs only 10/74 times better.
- Even if I ignore values with error estimate "error<1km" then CBG performs 32/74 times better, improved trilateration performs 10/74 times better and the rest are unaccounted for.

Procedure to create CBG trilateration and multilateration comparison

First the target files need to be sorted according to the RTT in order to have better landmarks at the top of the file for every target. This creates no impact on multilateration but it helps trilateration achieve better results.

Second step is to generate a CSV file containing results for both trilateration and multilateration for all targets. The format is explained below. The format is also printed inside the script.

Format for CBG trilat/multilat CSV file

Hostname, Target IP, Act. Lat, Act. Long, Est. Lat(trilat), Est. Long(trilat), Land1Hostname(trilat), Land1Lat(trilat), Land1Long(trilat), Land1Dist(trilat), Land2Hostname(trilat), Land2Lat(trilat), Land2Long(trilat), Land2Dist(trilat), Land3Hostname(trilat), Land3Lat(trilat), Land3Long(trilat), Land3Dist(trilat), Error (km)(trilat), Est. Lat(multlat), Est. Long(multlat), Land1Hostname(multlat), Land1Lat(multlat), Land1Long(multlat), Land1Dist(multlat), Land2Hostname(multlat), Land2Lat(multlat), Land2Long(multlat), Land2Dist(multlat), Land3Hostname(multlat), Land3Lat(multlat), Land3Long(multlat), Land3Dist(multlat), Land4Hostname(multlat), Land4Lat(multlat), Land4Long(multlat), Land4Dist(multlat), Land5Hostname(multlat), Land5Lat(multlat), Land5Long(multlat), Land5Dist(multlat), Error (km)(multilat)

For understandability view the same format below:

Format for CBG trilat/multilat CSV file

```
#format required:
# requisite    source (line, column)
# ---------    ---------------------
# Hostname     Node_info.txt (x, 2)
# IP           @names
# Act. Lat     target file (0, 1)
# Act. Long    target file (0, 2)
#
# Est. Lat     .trib/.tribe/.tribef file (2, 1)
# Est. Long    .trib/.tribe/.tribef file (2, 2)
# L1 hostname  Node_info.txt (x, 2)
# L1 Lat       target file (1, 1)
# L1 Long      target file (1, 2)
# L1 Rtt       target file (1, 3)
# L2 hostname  Node_info.txt (x, 2)
# L2 Lat       target file (2, 1)
# L2 Long      target file (2, 2)
# L2 Rtt       target file (2, 3)
# L3 hostname  Node_info.txt (x, 2)
# L3 Lat       target file (3, 1)
# L3 Long      target file (3, 2)
# L3 Rtt       target file (3, 3)
# Error        .trib/.tribe/.tribef file (1, 2)
#
# Est. Lat     .cbgb file (2, 1)
# Est. Long    .cbgb file (2, 2)
# L1 hostname  Node_info.txt (x, 2)
# L1 Lat       target file (1, 1)
# L1 Long      target file (1, 2)
# L1 Rtt       target file (1, 3)
# L2 hostname  Node_info.txt (x, 2)
# L2 Lat       target file (2, 1)
# L2 Long      target file (2, 2)
# L2 Rtt       target file (2, 3)
# L3 hostname  Node_info.txt (x, 2)
# L3 Lat       target file (3, 1)
# L3 Long      target file (3, 2)
# L3 Rtt       target file (3, 3)
# L4 hostname  Node_info.txt (x, 2)
# L4 Lat       target file (4, 1)
# L4 Long      target file (4, 2)
# L4 Rtt       target file (4, 3)
# L5 hostname  Node_info.txt (x, 2)
# L5 Lat       target file (5, 1)
# L5 Long      target file (5, 2)
# L5 Rtt       target file (5, 3)
# Error        .cbgb file (1, 2)
#
# note: line index starts from 0, column index starts from 1
```

The scripts are listed below.

File	Description
cbgb files	These contain results for CBG multilateration via CBG bestline technique.
target files	These contain target files.
trib files	These contain results for CBG trilateration via CBG bestline technique.
CreateCSVDistance.pl	Script to generate outputDistance.csv file containing results for both multilateration and trilateration.
TargetFileSortRtt.pl	Sort target files based on RTT.
outputDistance.csv	The output file generated by the CreateCSVDistance.pl script. The format of the output is shown above.