NIST TIP

Cyber-Human-Infrastructure Interaction Demo

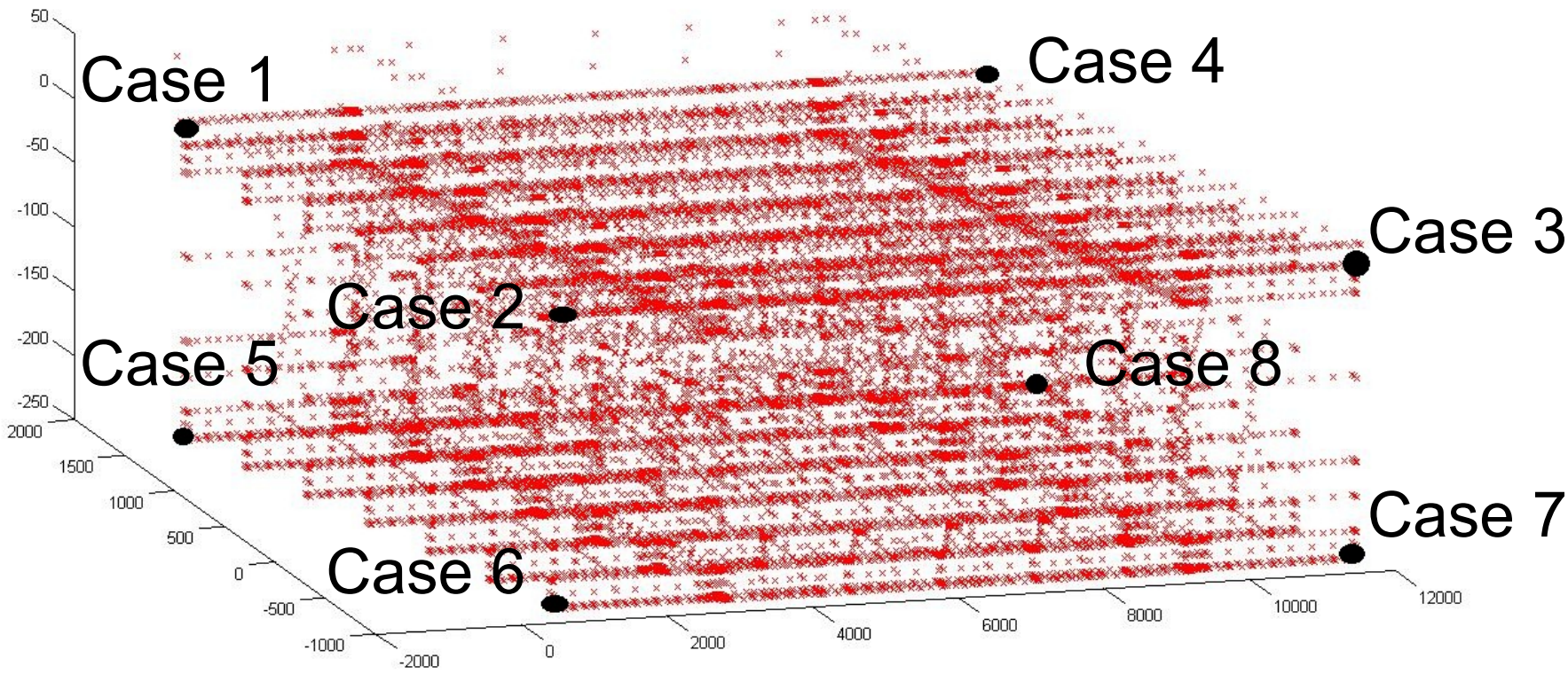
by

Manu Akula and Atul Sandur

04-07-2011

SPATIAL-CONTEXT AS INTERPRETED BY MATLAB CODE - THE "TRUE" SPATIAL-CONTEXT

All co-ordinates are in BFR; Image is x50 times



Inspector Position: Case X Far plane center: 0,0,0

Near plane center: Mid-point of Inspector Position and Far plane center Theta (Half angel) = 30º

SPATIAL-CONTEXT AS INTERPRETED BY MATLAB CODE - THE "TRUE" SPATIAL-CONTEXT

Nodes that are captured by the viewing cone (truncated) are documented in the following files:

1) result_matlab_1
2) result_matlab_2
3) result_matlab_3
4) result_matlab_4
5) result_matlab_5
6) result_matlab_6
7) result_matlab_7
8) result_matlab_8

The format of the result_matlab_X file is as follows:

NodeID X Y Z

CONVERTING BFR DATA INTO GPS AND IFR COORDINATES - The "true" GPS and IFR Coordinates

The 4 Points for calibration GPS coordinates (and corresponding BFR coordinates)

- We took 4 random GPS points in Google Earth.
- We designated one of the GPS points to an approximate BFR.
- We calculated the rest of the BFRs using Vincenty's Inverse.
- Even if there is an error, the error is translated for all points.

We calculated the Inspector's GPS coordinates for each of the 8 cases through Vincenty's Direct algorithm.

We determined the Roll, Pitch, Yaw using Auto-CAD.

We calculated the IFR of the far plane center (and near plane center) using Vincenty's Inverse algorithm.

SPATIAL-CONTEXT AS INTERPRETED BY JAVA CODE

The MATLAB BFR points were converted into IFR and GPS using Vincenty's (Direct and Inverse) algorithms and are fed into the Java Code.

The Java Code then re-converts the IFR and GPS coordinates into BFR coordinates using a Localized Linear Approximate Transformation Algorithm for GPS.

The Java Code then determines the list of points contained within the truncated cone.

NOTE: Had the Java Code used Vincenty's (Inverse and Direct) to re-convert the points (which were previously converted via Vincenty's Direct and Inverse) then the list would be identical to the Matlab List for each of the 8 cases.

SPATIAL-CONTEXT AS INTERPRETED BY JAVA CODE

BUT the Java Code re-converts the IFR and GPS coordinates into BFR coordinates using a Localized Linear Approximate Transformation Algorithm for GPS.

This means that there would be a discrepancy between the two lists (result_matlab_X and result_java_X).

This discrepancy is caused due to the approximations and assumptions used in the Localized Linear Approximate Transformation Algorithm in the Java code.

The discrepancy can be determined by comparing the two lists using the comparison code described in the following slide.

SPATIAL-CONTEXT INTERPRETATION LISTS COMPARISON CODE - MATLAB vs JAVA LISTS

result_matlab_1

NodeID X Y Z

result_java_1

NodeID X Y Z

Exhaustive hash index search across both lists.

3 statistical factors are determined.

- SF1: Number of common nodes (Higher the better)
- SF2: Number of nodes in matlab but not in java (Lower the better)
- SF3: Number of nodes in java but not in matlab (Lower the better)
- Corresponding %s

SPATIAL-CONTEXT INTERPRETATION LISTS COMPARISON RESULTS - MATLAB vs JAVA LISTS

Case	SF1 %	SF2 %	SF3 %
1	99.66	0.34	0.34
2	98.87	1.13	1.12
3	99.93	0.07	0.013
4	99.99	0.01	0.013
5	99.67	0.33	0.33
6	98.90	1.10	1.10
7	99.95	0.05	0.013
8	99.99	0.013	0.013

Vincenty vs Localized Linear

ase #	Vin. SF1	Vin. SF2	Vin. SF3	SF1 %	SF2 %	SF3 %
1	99.66	0.34	0.34	99.659	0.3401	0.3401
2	98.876	1.124	1.124	98.876	1.124	1.124
3	99.986	0.0134	0.0534	99.933	0.0667	0.0134
4	99.987	0.01302	0.01302	99.987	0.01302	0.01302
5	99.665	0.334	0.667	99.667	0.333	0.333
6	98.901	1.099	1.099	98.901	1.099	1.099
7	99.987	0.0134	0.0534	99.947	0.0534	0.0133
8	99.987	0.01302	0.01302	99.987	0.01302	0.01302

Code Running Time: Vincenty vs Localized Linear

$Vin.Time_0$ = 0 nano seconds (no fixed calibration)

$Vin.Time_1$ = Time taken for computing the following:

IO_BFR, IFROrient_BFR, FPD, NPD

$Vin.Time_2$ = Time taken to run through 17k nodes

$LL.Time_0$ = Time taken for computing the following:

BGMatrix, ScalingFactors, BO

$LL.Time_1$ = Time taken to compute bounding box parameters

$LL.Time_2$ = Time taken to run through 17k nodes

Average Run Times for the 8 cases in milli-seconds

Case #	LL.T0	LL.T1	LL.T2	Vin.T1	Vin.T2
1	18.13	0.237	360.64	18.53	361.13
2	17.45	0.221	347.72	18.68	339.54
3	18.09	0.270	458.27	20.85	460.03
4	17.53	0.237	460.18	19.90	462.11
5	17.64	0.231	366.90	20.08	365.80
6	17.66	0.282	337.30	19.82	352.09
7	17.49	0.314	456.00	19.04	463.88
8	17.82	0.302	462.27	19.85	456.48