Student Satellite Project
Indian Institute of Technology, Bombay
Powai, Mumbai - 400076, INDIA

Website: www.aero.iitb.ac.in/satlab

README - Sky Coverage Calculation

Guidance, Navigation and Controls Subsystem

op_gnrt_sky_coverage_T.m

Code Type: MATLAB - Function
Code author: KT Prajwal Prathiksh
Created on: 13/01/2021
Last modified: –/–/—-
Reviewed by: NOT YET REVIEWED!
Description:
This function generates the table required for the calculation of the sky coverage, by calling the following function from the **Star Catalogue Preprocessing** block:

1. `ca_CartVect_2_RA_DE.m` - To convert the Cartesian boresight vectors to the Right-Ascension and Declination counterpart.

and the following functions from the **Processing** block of **Sensor Model** code:

1. `se_PR_1_Trim2FOV.m` - To trim the Star Catalogue to the Field-of-View of the sensor.
2. `se_PR_2a_ICRS2Lens.m` - To transform the Star Coordinates from the ICRS Frame to the Lens Frame.
3. `se_PR_3_Lens2Sensor.m` - To Transform the Star Coordinates from the Lens Frame to the Sensor Frame.
4. `se_PR_4_Trim2Sensor.m` - To remove the stars that lie beyond the boundary of the sensor.

The function takes in the input of all the boresight vectors (N_{BV}) which correspond to uniformly distributed direction vectors on the celestial sphere.
The function subsequently calculates N - number of images per boresight vector (`op_N_Images`) which correspond to N images with an angle of rotation about the boresight axis, which varies from $[0°, 360°]$ in increments of $360°/N$.
Therefore, the function has to iterate over $N_{BV} \times N$ images. The sky coverage is therefore estimated as the number of images which can detect stars greater than $N_{TH}/\alpha_{SM}\%$, divided by the total number of images.
Here N_{TH} is the minimum number of stars which are needed to be matched accurately by the Star Matching algorithm, such that the Estimation algorithm can estimate the attitude within the accuracy constraints of the system.
Here, the Star Matching algorithm can be characterised to have a matching accuracy (α_{SM}), which is not 100%. Therefore the sensor actually has to be able to detect stars equal to or greater than $N_{TH}/\alpha_{SM}\%$.

NOTE: The function uses the `parfor` command, and therefore requires the **Parallel Computing Toolbox** of MATLAB. The first time this code is run, MATLAB will take some time in creating the parallel pool workers. This is a one-time event that occurs only during the first run. Therefore care must be taken to ensure that in the first-run the function does not iterate completely over the $N_{BV} \times N$, but instead something of the order of ≈ 10. The function subsequently needs to be executed around $3-4$ times till the run-time for the ≈ 10 iterations converge.
Once the run-time converges, the function can be run over all the $N_{BV} \times N$ images, and a speedup of $\approx 4\times$ can be expected on a Quad-Core machine.
Formula & References: —
Input parameters:

1. **se_bo** : (Table)
2. **se_er** : (Structure)
3. **se_ig** : (Structure)
4. **se_in** : (Structure)
5. **se_op** : (Structure)
6. **se_T** : (Table)
7. **op_in** : (Structure) Contains all the variables required to calculate the sky coverage
   (a) **es_N_TH** : (Integer) Minimum number of stars required to be matched accurately to estimate the attitude within the accuracy requirements of the system.
   (b) **sm_matching_accuracy** : (Float) Accuracy of the Star Matching algorithm.
   (c) **op_N_Images** : (Integer) Number of images per boresight vector which correspond to N images with an angle of rotation about the boresight axis, which varies from $[0°, 360°]$ in increments of $360°/N$.
   (d) **iter** : (Integer) Number of iterations to be performed by the function. **Notes:** This is important particularly when running the code for the first-time, when it has to be set equal to 10. Once the code is optimized by MATLAB, set it equal to -1, which will execute the code over $N_{BV} \times N$ iterations.
   (e) **v_FOV** : ((110,3) - Matrix) Contains the Cartesian vectors for all 1100 uniformly distributed points over a sphere.

Output:

1. **op_sky_coverage_T** : (Table) The columns are as follows:
   (a) op_r0 - The Cartesian vector corresponding to the boresight vector.
   (b) Total_N - Total number of stars visible within the FOV.
   (c) Mean_Sensor_N - Mean number of stars which can be detected by the sensor in the `op_N_Images` number of images per boresight vector
   (d) STD_Sensor_N - Standard deviation in the number of stars which can be detected by the sensor in the `op_N_Images` number of images per boresight vector
   (e) Mode_Sensor_N - Mode of the number of stars which can be detected by the sensor in the `op_N_Images` number of images per boresight vector
   (f) Images_N - `op_N_Images`
   (g) Valid_Images_N - Number of images which had stars greater than $N_{TH}/\alpha_{SM}\%$
2. **op_sky_coverage_val** : (Float) The sky coverage value in terms of percentage

op_sky_coverage_main.m.m

Code Type: MATLAB - Function
Code author: KT Prajwal Prathiksh
Created on: 13/01/2021
Last modified: –/–/—-
Reviewed by: NOT YET REVIEWED!
Description:
This is the main script which is used to calculate the sky coverage value of the system.
Formula & References: —
Input parameters:

1. **op_in** : (Structure) Contains all the variables required to calculate the sky coverage
   (a) **es_N_TH** : (Integer) Minimum number of stars required to be matched accurately to estimate the attitude within the accuracy requirements of the system.
   (b) **sm_matching_accuracy** : (Float) Accuracy of the Star Matching algorithm.
   (c) **op_N_Images** : (Integer) Number of images per boresight vector which correspond to N images with an angle of rotation about the boresight axis, which varies from $[0°, 360°]$ in increments of $360°/N$.
   (d) **iter** : (Integer) Number of iterations to be performed by the function. **Notes:** This is important particularly when running the code for the first-time, when it has to be set equal to 10. Once the code is optimized by MATLAB, set it equal to -1, which will execute the code over $N_{BV} \times N$ iterations.
   (e) **v_FOV** : ((110,3) - Matrix) Contains the Cartesian vectors for all 1100 uniformly distributed points over a sphere.

Output: Writes the op_sky_coverage_table.csv, and saves the variables: `op_sky_coverage_T`, `op_sky_coverage_val` in `./Optics/Output/` directory.

ca_CartVect_2_RA_DE.m

Code Type: MATLAB - Function
Code author: KT Prajwal Prathiksh
Created on: 13/01/2021
Last modified: –/–/—-
Reviewed by: NOT YET REVIEWED!
Description:
This function converts a given unit vector in a rectilinear coordinate system - (X, Y, Z) to Right-Ascension and Declination components. The (X, Y, Z) coordinate system definition corresponds to the projection of the Earth' North Pole onto the celestial sphere as the Z-axis, and the vernal equinox as the X-axis,at epoch ICRS2000, with the Y-axis completing the right-handed orthonormal coordinate system:

$$Z = X \times Y$$

Formula & References:
Formula: For input $[X, Y, Z]$, output is:

$$RA = atan2(Y, X)$$

$$DE = asin(Z)$$

Because:

$$X = \cos(RA)\cos(DE)$$

$$Y = \sin(RA)\cos(DE)$$

$$Z = \sin(DE)$$

Reference:

1. Refer 4.1.1 - Computer Science Corporation, *"SKYMAP Requirements, Functional, and Mathematical Specifications, Volume 3, Revision 3"*. (1999).

Input parameters:

1. **X** : (Float) - X-component of the unit vector
2. **Y** : (Float) - Y-component of the unit vector
3. **Z** : (Float) - Z-component of the unit vector

Output:

1. **RA** : (Float) - Right-Ascension component - in degrees
2. **DE** : (Float) - Declination component - in degrees