Deepspace Interplanetary Navigation Operations Colorado Research EXplorer (DINO C-REx)

DINO C-REx Technical Memorandum

Document ID: DINO_C-REx-Image Generation

CAMERA OBJECT UNIT TESTS

Prepared by	Matt Muszynski

Status: Initial Version
Scope/Contents
Description of unit tests completed for the DINO C-REx camera object. This includes tests 4.8 through 4.18.

Rev:	**Change Description**	**By**
1.0	Initial Release	Matt Muszynski

Contents

1 Overview

This document describes all unit tests written for the image object by the DINO C-REx camera model team. Each method of the image object has an associated test, and additional integrated tests are performed where deemed useful by the team. Each test has an entry in this document for status (including date of status change), responsible DINO C-REx team member, a qualitative description of the test performed and a brief technical discussion of the method employed in code to achieve the test. Because it is used in the image.update_state() method, tests for em.planck() are also recorded in this document.

2 Tests

2.1 Test 4.8: image.find_stars_in_FOV()

Status: Test Passes. 12.1.17
Responsible Team Member: Matt Muszynski
Description: This method is responsible for limiting objects to only those in the field of view of the camera. Here, an image calculated at run time is compared to known star catalogs to ensure all expected stars (above a reasonable magnitude) are present.
Method: For this test, two starfields are calculated. The stars present are matched to known starfields manually, and satisfied that they are displaying the correct stars, RA and DE are saved to .npz save files. When the test is run, the RA and DE values for each image calculated at run time are compared with those saved in the npz files. See the appendex below for a listing of the exact stars checked.

- Image 1: Orion
  - Detector Height: 1.5 cm
  - Detector Width: 1.5 cm
  - Focal Length: 3.0 cm
  - Maximum Magnitude: 3.57
  - Boresite RA: 85°
  - Boresite DE: 0°
  - Boresite Twist Angle: 0°
- Image 2: Ursa Minor
  - Detector Height: 2.3 cm
  - Detector Width: 2.3 cm
  - Focal Length: 4.0 cm
  - Maximum Magnitude: 4.0
  - Boresite RA: 187°
  - Boresite DE: 59°
  - Boresite Twist Angle: 0°

2.2 Test 4.9: image.remove_occulatations()

Status: Test Passes. 11.22.17
Responsible Team Member: Matt Muszynski
Description: To test this method, an extended spherical body is placed in the center of a frame. We then check all stars that should be present in that frame and ensure that all objects closer to the center of the frame than half the object's angular size are appropriately removed.
Method:

1. Set up the test.
   (a) Set the position of a body (named Earth) to be $\vec{r}_{earth} = [1au, 0, 0]$, the position of the spacecraft to be $\vec{r}_{s/c} = [1au - 250000km, 0, 0]$, and give the spacecraft an attitude DCM of $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$. This puts the spacecraft 250,000 km from the Earth pointing so that the earth is at the exact center of its field of view. Because the camera to body DCM is also an identity matrix, the camera points exactly where the spacecraft does.
   (b) Set the camera debug messages such that occulted stars are removed, but the resolved body is not added.
   (c) Take an image with a single frame by setting taekImage to 1, running starCam.updateState(), setting takeImage to 0, and running starCam.updateState() one more time.
   (d) Set the camera debug messages such that occulted stars are not removed, and the resolved body is not added.
   (e) Take an image with a single frame by setting taekImage to 1, running starCam.updateState(), setting takeImage to 0, and running starCam.updateState() one more time. Now we have two images, one with all the stars in the FOV including those that are occulted, and one without those that are occulted.
2. Test that no non-occulted stars are closer to the center of the FOV than the limb of the earth.
   (a) For every non-occulted star, find the distance between the center of the FOV and where it appears on the focal plane.
   (b) Project phsical distance into angular distance
   (c) Calculate angular distance between the center of the FOV and the limb of the earth.
   (d) Assert that none of the non-occulted stars are closer to the center of the FOV than the limb of the earth.
3. Test that no occulted stars are farther to the center of the FOV than the limb of the earth.
   (a) Compare the image above with all stars with the image with only non-occulted stars. Use this to create a set of only occulted stars.
   (b) Assert that the lengths of the array containing occulted stars and the array containing non-occulted stars sum to the length of the array of all stars from image where occulted stars were not removed.
   (c) Find physial distance between center of FOV and each occulted star.
   (d) Project that physical distance into angular distance in the FOV.
   (e) Assert that none of the non-occulted stars are farther from the center of the FOV than the limb of the earth.

Fig. 1: Best case plots show many solutions for 512×512 camera.

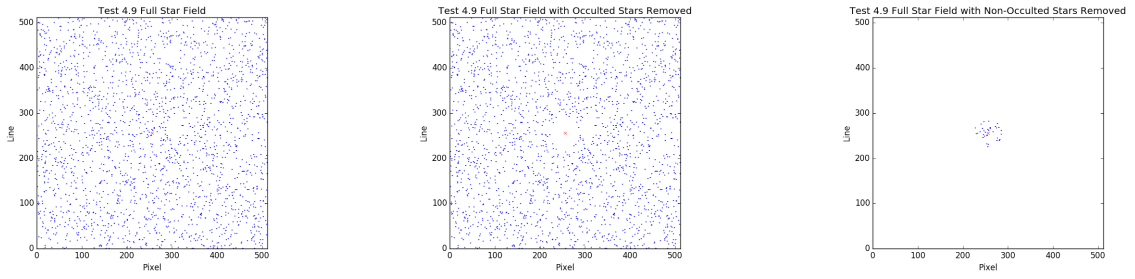

Fig. 2: All stars used in test 4.9. Images show full star field, set of non-occulted stars, and set of occulted stars.

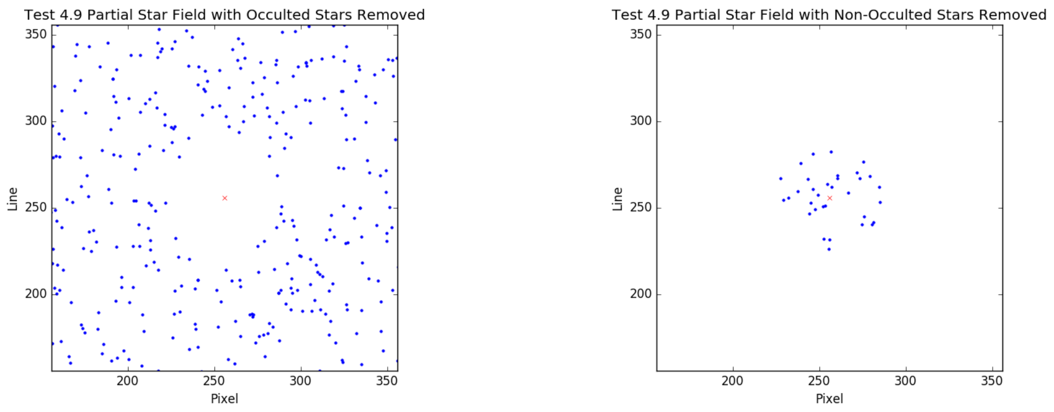

Fig. 3: Zoomed images of test 4.9 star field. Images show set of non-occulted stars and set of occulted stars.

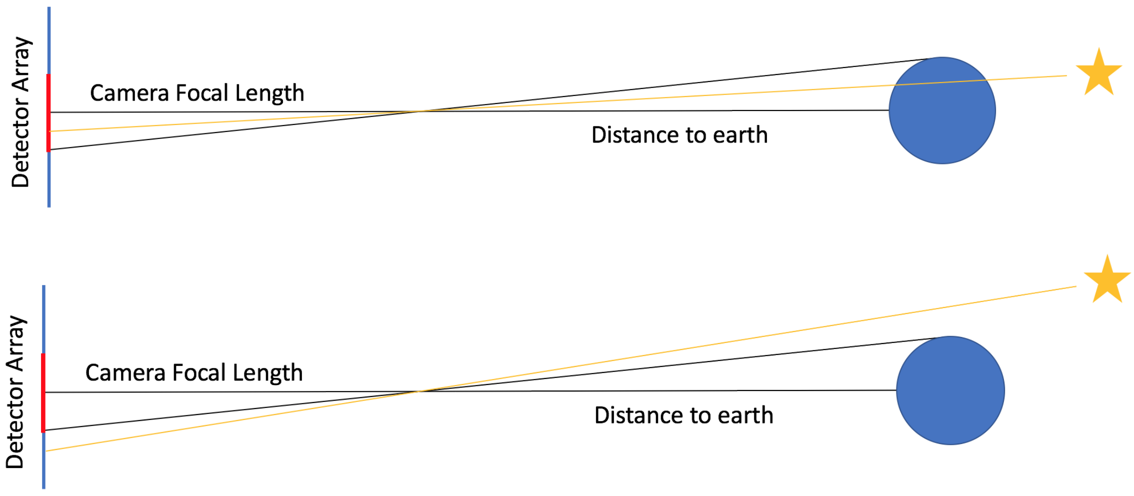

Fig. 4: Cartoon showing the mapping of occulted and non-occulted stars onto the detector plane (blue). We see that the occuted star appears within the extended image of the Earth on the detector plane (red), and that the non-occulted star appears outside of the image of the Earth.

2.3 Test 4.10: image.psf()

Status: Passes. 12.2.17
Responsible Team Member: Matt Muszynski

Description: Because we expect the gaussian PDF applied to stars and body facets to be normalized, this test checks that the sum of the PSF caluclated by image.psf() is indeed one.
Method: This test places a single at the center of an image using a camera with a very small FOV (called tinyCam). There are no other stars in the image. All intensity values in the image are summed and compared to the intensity value of the star without PSF applied. The test assers that the two match.

2.4 Test 4.11: image.psf()

Status: Passes. 12.2.17
Responsible Team Member: Matt Muszynski
Description: The PSF calculated at runtime should be centered at (0,0) in pixel/line space. This test ensures that it is.
Method: This test uses the same tinyCam image as test 4.10. Here, the average pixel and line position of all PSF star facets is computed and compared to (256,256), the center of the FOV. The test asserts that the average matches the true position to within machine precision.

2.5 Test 4.12: image.psf()

Status: Passes. 12.03.17
Responsible Team Member: Matt Muszynski
Description: This test fits a 2D gaussian to the PSF created my image.psf() at run time to enure the correct covariance is achieved.
Method: In this test, a bivariate normal function is sampled at all the same pixel and line coordinates as are delivered by the camera. A normalization factor is applied to give the two the same amplitude, and the difference between the two results is checked to within maching precition. Since this test only cares about checking the standard distribution of the distribution, amplitude does not matter, and the normalization factor is appropriately applied.

2.6 Test 4.13: image.rasterize()

Status: Passes. 12.3.17
Responsible Team Member: Matt Muszynski
Description: Check that the flooring and binning of image.rasterize() is funcitoning as expected.
Method: This takes 10 randomly generated pixel, line, and intensity values and compares a rasterized array generated from them at run time with one that was calculated by hand offline. Because the final values are integers, the test asserts that the array calculated offline matches the array calculated at runtime exactly.

2.7 Test 4.14: image.add_read_noise()

Status: Incomplete. 10.31.17
Responsible Team Member: Ishaan Patel
Description: This test adds read noise to a blank image and ensures that it is in fact gaussian as desired.
Method:

2.8 Test 4.15: image.add_poisson_noise()

Status: Incomplete. 10.31.17
Responsible Team Member: Ishaan Patel
Description: This test adds poisson noise to a false detector array and checks the output to ensure it is truly a poisson distribution.
Method:

2.9 Test 4.16: Image Pixel and Line Conversion

Status: Test Passes. 11.22.17
Responsible Team Member: Matt Muszynski
Description: This method takes all stars from a scene, calculates their angular separation using their pixel and line coordinates and the physical characteristics of the camera that took the image. This is then compared to the true angular separation between the two stars using the dot product of the unit vector pointing at each.
Method: Test 4.16 leverages the same star field as test 4.9.

1. Compute the distance between the center of the field of view and each star, both in the pixel direction and the line direction.
2. Calculate the physical size of each pixel in pixel and line directions (physical size of the detector in that direction divided by the number of pixels in that direction)
3. Compute the distance between the center of the field of view and each star on the detector plane in physical distance in pixel and line directions (multiply pixel distance by physical size in the pixel direction and line distance times the physical size of a pixel in the line direction).
4. Calculate the true distance between the center of the FOV and each star in physical space (using Pythagorean theorem).
5. Calculate the angular distance between the center of the FOV and each star by projecting using the physical distance between the two points and the focal length.
6. Calculate the true angular distance between the center of the camera FOV and the star using dot products.
7. Assert that both angular distances are the same.

2.10 Test 4.17: em.planck()

Status: Test Passes. 10.31.17
Responsible Team Member: Matt Muszynski
Description: This test checks that the value of the integral of em.planck() over all wavelength space to the output of em.stefan_boltzmann() matches to within 0.1%
Method: This method performs a numerical integration of the planck functon over a wavelength range of 1nm to 10000nm in 1nm bins. Values outside of this range are ignored as they are very small. This involves evaluating em.planck() at all wavelengths and multiplying by 1nm (essentially a Riemann sum). This then gives a value in units of $W/m^2/sr$. The sr portion of this value can be removed by multiplying by a factor of pi (the solid angle subtended by the source as measured at the source). Theoretically this value should match the output of the stefan-boltzmann function exactly. In order to allow for the error inherent in the numerical integration, we only assert that the values match to within 0.1%.

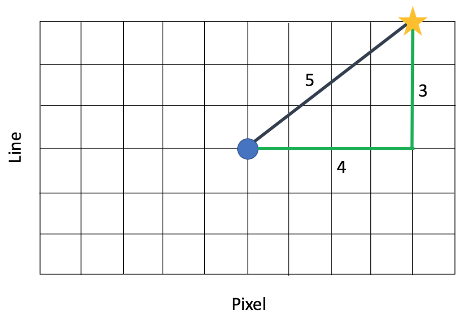

Fig. 5: Here, the star appears 3 lines and 4 pixels away from the center of the FOV (blue dot). Assuming each pixel is 1 unit by 1 unit, this makes the star 5 units away from the center of the FOV. Test 4.16 shows that it does.

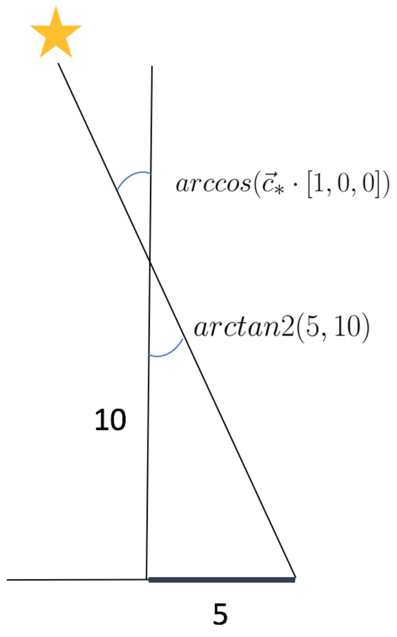

Fig. 6: Assuming that the focal length of the camera is 10 units, we can use arctan2 to find that the angular distance between the center of the FOV and the star must be 30 degrees. This must match the arccos of the dot product between the star's unit vector $\vec{c}_*$ and the boresite vector in camera coordinates, $[1, 0, 0]$

2.11 Test 4.18: em.planck()

Status: Test Passes. 10.31.17
Responsible Team Member: Matt Muszynski
Pytest Function Name:
Description: This test checks that the value of the integral of em.planck() over all wavelength space to the accepted solar constant value of $1367 W/m^2$ to within 0.1%
Method: The method for test 4.18 matches that of 4.17 through multiplying by the solid angle subtended by the source, which here does not reduce to pi, but is rather $\pi \frac{r_\odot^2}{a_\male^2}$ where $a_\male$ is the semi-major axis of Earth's orbit and $r_\odot$ is the radius of the sun. This gives the value of solar flux as measured from 1au, which is well characterized to be 1367 W/m^2. Again, we test that the computed value matches the accepted emprical value to within 0.1%.

3 Appendix

Table 2: Manual Verificaiton List: Ursa Minor

Star Name	Expected RA	Actual RA	Expected DE	Actual DE
α UMi	37.9458	37.9466	89.2642	89.2641
β UMi	222.6750	222.6767	74.1556	74.1555
γ UMi	230.1833	230.1823	71.8339	71.8340
ϵ UMi	251.4917	251.4924	82.0372	82.0372
5 UMi	216.8833	216.8813	75.6958	75.6959
ξ UMi	236.0125	236.0144	77.7944	77.7945
δ UMi	263.0542	263.0537	86.5864	86.5863
RR UMi	224.3958	224.3963	65.9325	65.9324
4 UMi	212.2125	212.2125	77.5475	77.5474
η UMi	244.3792	244.3770	75.7547	75.7547

Table 3: Manual Verificaiton List: Orion

Star Name	Expected RA	Actual RA	Expected DE	Actual DE
β Ori	88.7929	88.7929	7.4070	7.4070
γ Ori	81.2828	81.2828	6.3497	6.3497
ϵ Ori	84.0534	84.0534	-1.2019	-1.2019
ξ Ori	85.1897	85.1897	-1.9426	-1.9426
κ Ori	86.9391	86.9391	-9.6696	-9.6696
δ Ori	83.0017	83.0017	-0.2991	-0.2991
ι Ori	83.8583	83.8583	-5.9099	-5.9099
η Ori	81.1193	81.119	-2.3971	-2.3971
λ Ori	83.7845	83.7845	9.9342	9.9342
τ Ori	79.4017	79.4017	-6.8444	-6.8444

3.1 Test 4.19: image.photonEnergy()

Status: Complete. 12.3.17
Responsible Team Member: Matt Muszynski
Description:
Method: This test compares the energies for a set of 10 wavelengths computed at runtime to the energies for the same wavelengths computed offline. The test asserts that they match to within machine precision.