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

DINO C-REx Technical Memorandum

Document ID: DINO_C-REx-Image Generation

SYSTEMS ENGINEERING REPORT 4.3A: METHOD FOR COLLECTING TC AND QE DATA.

Prepared by	Matt Muszynski

Status: Initial Version
Scope/Contents
User documentation for the DINO C-REx Camera Module.

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

Contents

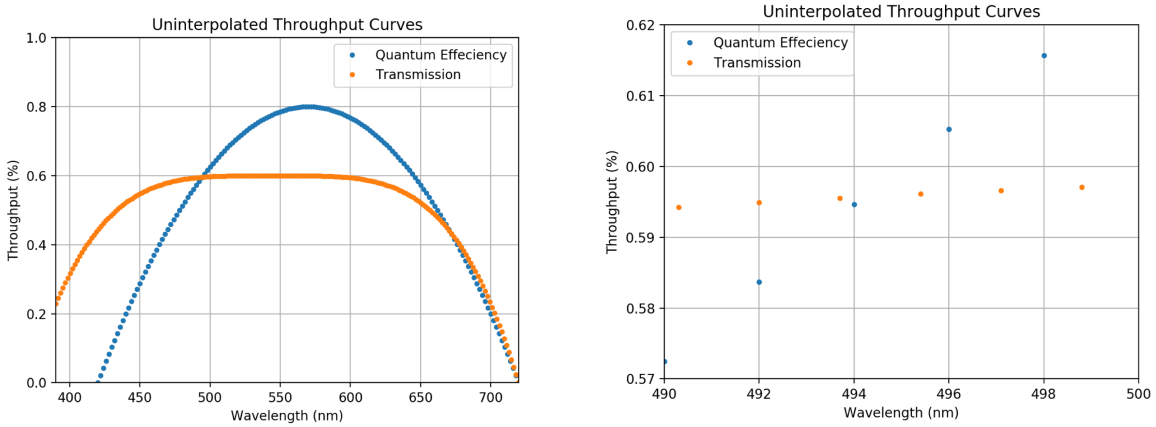

Fig. 1: Example quantum efficiency and transmission curves, sampled as discrete values. Left image shows all points defined, and right image shows zoomed view of 490 to 500 nm region. Note that the QE curve (blue) is sampled every 1 nm, while the transmission curve is sampled every 0.7 nm. The curves will need to be unified before their respective throughput arrays can be multiplied together to create a sensitivity curve.

1 Overview

To fulfill the requirement of simulating a camera's bandpass, the camera team decided to go with a relatively high fidelity model, simulating the interaction between the transmission properties of a camera lens and the quantum efficiency properties of a camera CCD. Both are input by the DINO C-REx user when initializing the camera, but because they are not trivial to set up, this document and its partner (SER 4.3b) illustrate how to set up transmission and quantum efficiency curves within DINO C-REx and how they are used by the system.

2 Basic Implementation

Both transmission and quantum efficiency curves are inputs to the DINO C-REx camera model, and both are passed as python dictionaries. This allows the user to quickly and easily capture the two-dimensional aspect of transmission and quantum efficiency curves (i.e. which wavelengths the curves are sampled at and the throughput values of the curve at those wavelengths).
Each dictionary should have two entries, each indexed with a string. One entry should be indexed with the string 'lambda', and should contain wavelenghth values (in nm) as a 1xn numpy array. The second entry into each dictionary is 'throughput', and should be a 1xn numpy array with either percent throughput at each of the wavelengths in the 'lambda' array (for transmission curves) or a photon to electron quantum efficiency conversion (for quantum efficiency). Importantly, the 'lambda' entries of the two dictionaries do **NOT** need to match as that will be handled by the camera module.

3 Calculation of Sensitivity Curve

When the camera object is initialized, the QE and transmission curves entered by the user will be combined to create 'sensitivity curve', a set of throughput values that can do the job of both transmission and quantum efficiency curves, converting Power at each wavelength straight into electron flux due to photons at that wavelength coming off of the camera detector. See fig. 1.

3.1 camera.findLambdaSet()

The first step of unifying the curves is to select the set of wavelength values at which the final sensitivity curve will be sampled. This is done by camera.findLambdaSet(). This function finds the smallest and largest wavelengths represented in either the QE or transmission curve, and creates a set of evenly spaced wavelength bins between them (at the width specified by the user in the variable lambdaBinSize). For

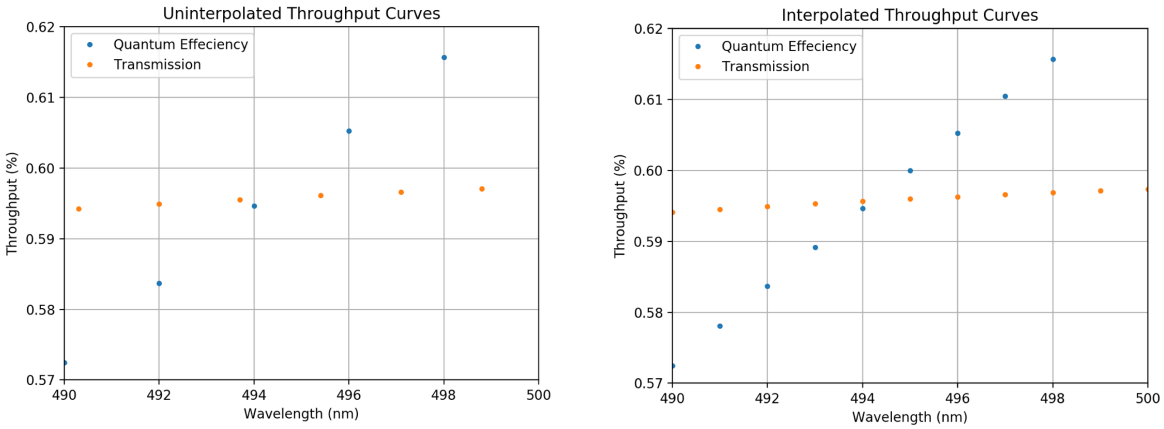

Fig. 2: The zoomed plot from above, paired with the same range after unification. Notice that after camera.interpolateLambdaDependent() is run, both QE and transmission curves are sampled at exactly 0.5 nm intervals. This allows the two to be simply multiplied together to find a sensitivity curve.

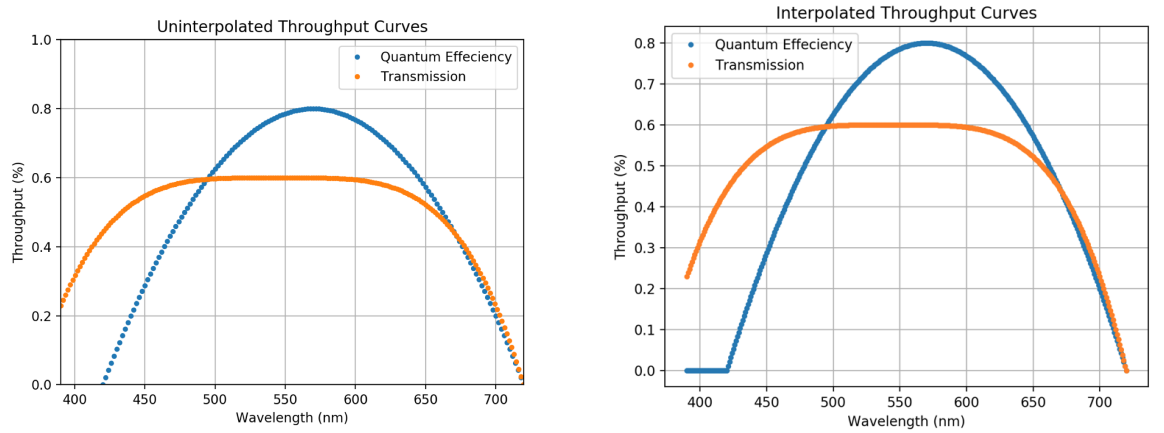

Fig. 3: The full quantum efficiency and transmission curves. Notice that the QE curve now has values below the minimum set by the user (420 nm), but that throughput values have been set to zero.

the example in figure 1, this runs from 390nm to 740nm, and has bins spaced by 0.5 nm.

3.2 camera.interpolateLambdaDependent()

Next, camera.interpolateLambdaDependent() is used to calculate throughput values between those set by the user in the dictionaries provided. Note that the lambdaSet camera.findLambdaSet() produced will have many entries not represented in the lambda arrays in the dictionaries provided by the user. This function walks through the values in lambdaSet. Where a wavelength value in lambdaSet was provided by the user in one of the dictionaries, that throughput value is used for the interpolated throughput. If there is no matching value, camera.interpolateLambdaDependent() finds the throughput values of the wavelengths in the user provided dictionaries that are immediately to the left and right of the desired value and calculates a new throughput value via linear interpolation. If there is no value to the right or left (i.e. the desired value is higher or lower every entry in the user provided array), throughput is set to zero.