Satellite Filter Project: Camera State Model Report

Sibley School of Mechanical and Aerospace Engineering
Spring 2022
Evan Wilt
ew399@cornell.edu

Abstract

This project is composed of two parts: A standard STOP (structural, thermal, optical, performance) analysis done in Ansys Workbench, and data fit to a camera distortion model of a Higher Order Unscented Filter (HOUSE). It utilizes incident radiation calculations to simulate induced thermal stress and deformation. The deformed lens geometries can be analyzed in Ansys Speos via tracing ray impacts on an irradiance sensor. The difference in irradiative patterns on this sensor, combined with comparisons of the deformed lens mesh to its undeformed version, inform the calculation of the desired Zernike polynomials (measures of spherical aberration). Having acquired these polynomials, a discretized radial distortion map can be calculated for a given lens (of a given radius and focal length). The distortion map for each timestep can then be fit to the filter's camera distortion model. At the time of writing this report, the data fitting is incomplete but estimated to be finished by Summer 2022.

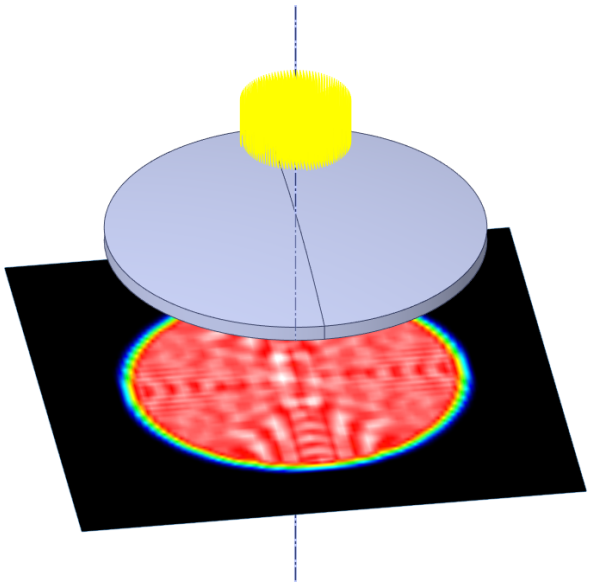

Figure 1: Results of a Ray Tracing Simulation

Table of Contents

Introduction

A significant portion of this project is made possible by advancements in commercial FEA, optics, and ray tracing software. Ansys Inc has been incredibly cooperative in helping apply and tweak their software to the problems faced here. Specifically, in their transient thermal, structural, and optical programs. With the incorporation of trajectory data from the Systems Toolkit (STK), every step of this project could be performed entirely within the Ansys Ecosystem (this allows for ease of updates in model parameters). However, much of this project was also made possible by custom scripting through the IronPython implementation in workbench. Though it is highly efficient for automation of this project, its documentation is unclear at times. Where possible, this report attempts to make clear exactly how to recreate the necessary automation for similar projects, but the author's contact information has also been provided for any further questions the reader may have.

Thermal and Structural Modeling

Simple Cube, Thermal

For easy validation of physical results, we first analyze a cube composed of six aluminum with five aluminum side panels and an aluminosilcate "camera" faceplate. Additionally, there is a model electronics tray in the center of the geometry that, at this point in the process, does not have internal heat generation modeled (this is included in the updated geometry).

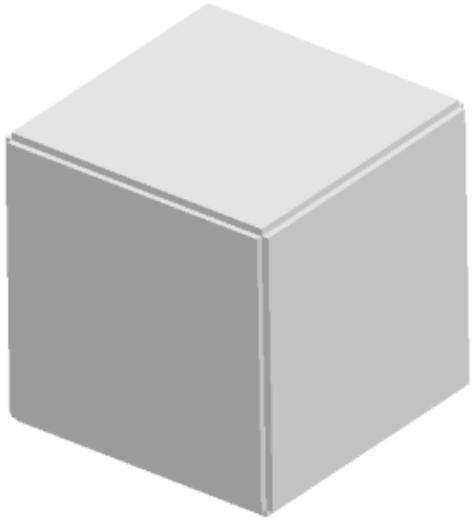

Figure 2: Simple Satellite Geometry

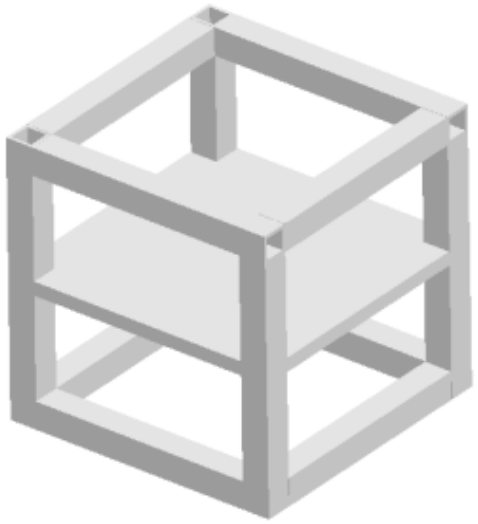

Figure 3: Simplified Satellite Internals

Transient Structural (With Lens)

At this step, the lens is introduced into the geometry. The CAD is taken from the thorlabs catalog and is a 75mm diameter, UV Fused Silica Plano-Convex Uncoated lens with a focal length of 501.8mm [1]. It is placed within the top plate in direct contact with the aluminum. To satisfy boundary conditions, a fixed displacement of 0 was applied at the center of the electronics plate. We can see that the stress around the point is unrealistically high but does not spread more than a few elements away from the BC. For now, this will be ignored as it does not seem to be significantly impacting the model (stress concentrations return to 0 within half a millimeter). To correctly remediate this issue, the 3-2-1 method of constraints should be applied to future geometries. This method accurately creates zero stress boundary conditions for free floating models [2] .

Transient Thermal (With Lens)

Transient thermal analysis is again performed, this time with heat from the electronics tray included in the model. Efficiency of input power to the PCB is approximated at 40% (rough estimate for the moment). This did not seem to impact the heat at the walls significantly (hence the model validation performed to ensure that heat was in fact being transmitted through the contacts). It raises the overall temperature but not its distribution due to the symmetry of this load. There is significant uniformity of each wall's temperature. This is to be expected with the large displayed timesteps and uniform load on each wall. Looking at the values of each wall's

temperature shows slight variation from each other (as expected since each wall does not have the same magnitude of incident radiation).

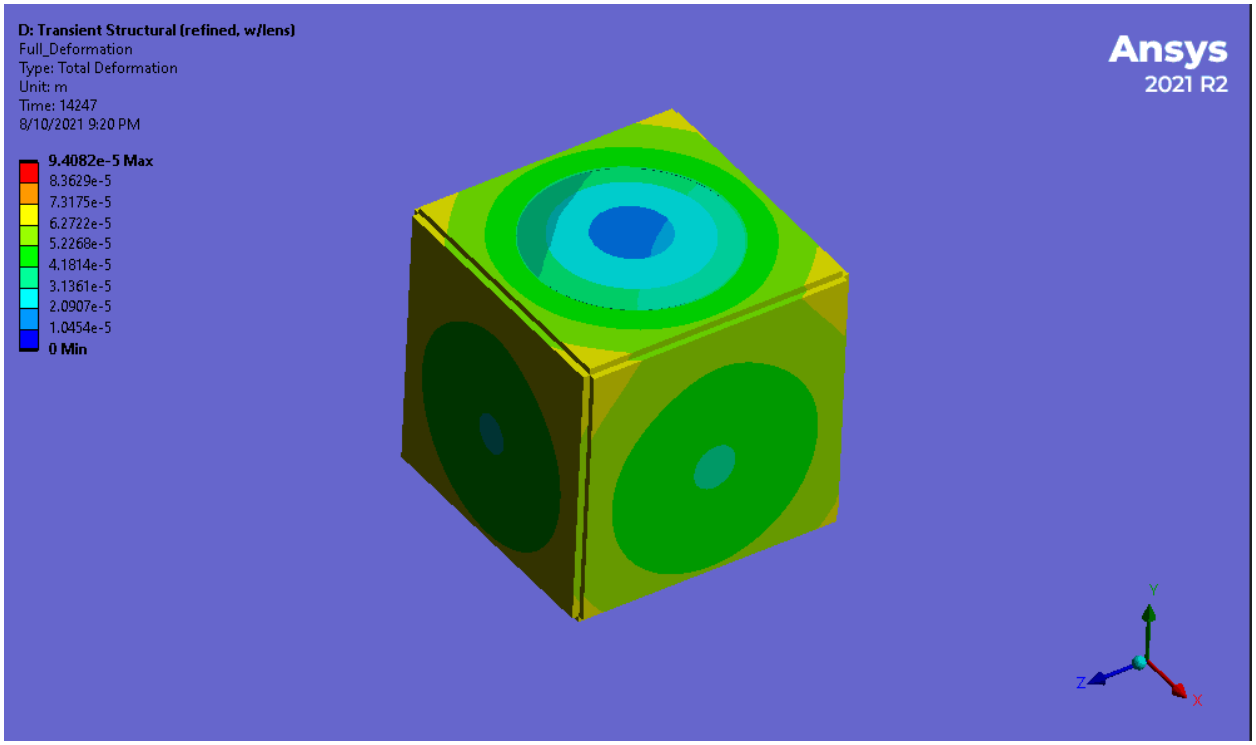

Figure 4: Transient Structural Deformation of Updated Geometry w/Lens

Transient Structural (With Lens, Refined)

This step provides the new structural results from the updated thermal model. Its main purpose is to format the data in such a way that it can be analyzed for its optical properties (Zernike coefficients). A custom script (fig 1 in the Appendix) processes each timestep to output an STL of the lens geometry at that step. Before this script can operate, the deformed timesteps must be expanded to be independently represented within the transient structural solution tree. Overall it covers 3 orbits of deformation. Each orbit, the deformation peak raises slightly more so it cannot be confirmed if it is in steady state or not. Deformations are on the scale of 1e-5 meters for the lens (significant enough to have optical impact). With the next geometry refinement, it is recommended to analyze more than 3 orbits of data until steady state patterns appear in thermal cycling. Notice how in fig 5, each thermal cycle reaches a slightly higher deformation peak (indicative that steady state has not been reached).

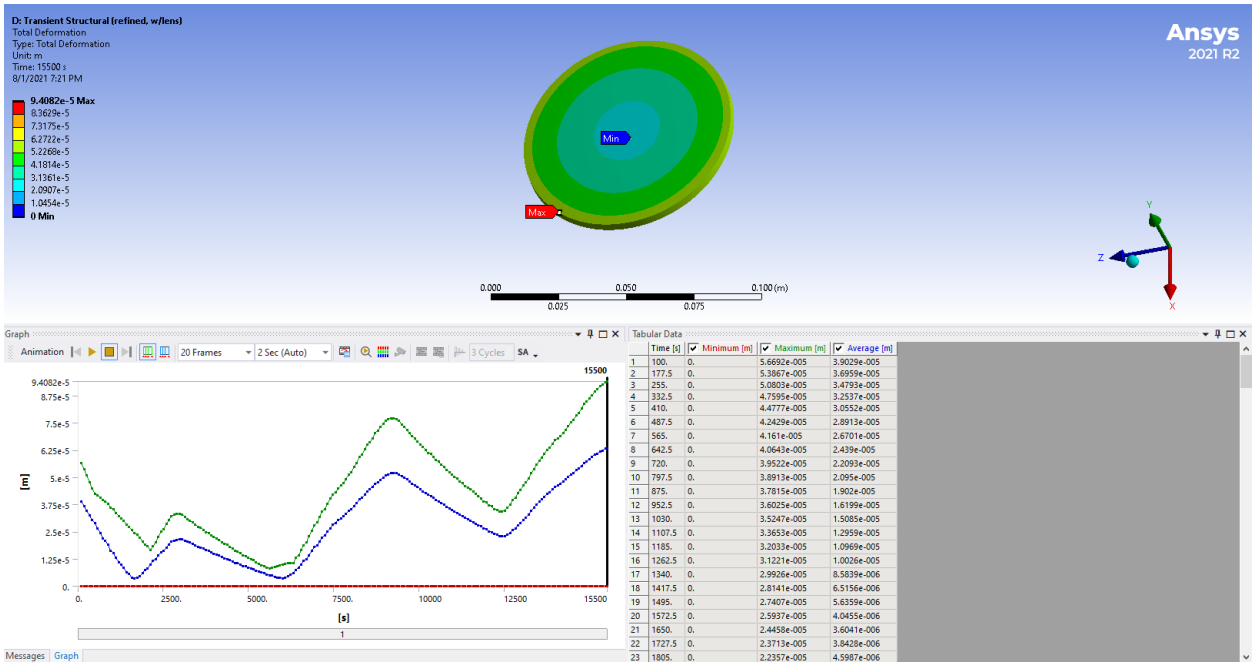

Figure 5: Lens Deformation Cycles For 3 Orbits

Optical Analysis (Speos)

SPEOS performs most analysis through ray tracing. In order to calculate the Zernike coefficients, it analyzes the difference between a deformed mesh, an undeformed mesh, and an LPF (light path file).To perform the simulation, a ray file is created that simulates the desired pattern (in this case circular), direction and number of rays to be simulated. It then simulates the path of the light through the geometry based on applied material properties (in this case a custom ideal lens that transmits 100% of rays).. The LPF's are generated from a sensor placed behind the warped geometry. Figs 2a through 6a depict the necessary settings for setup of the light simulation. After this point, it is best to utilize a script to analyze the large number of timesteps output from the transient structural simulation.

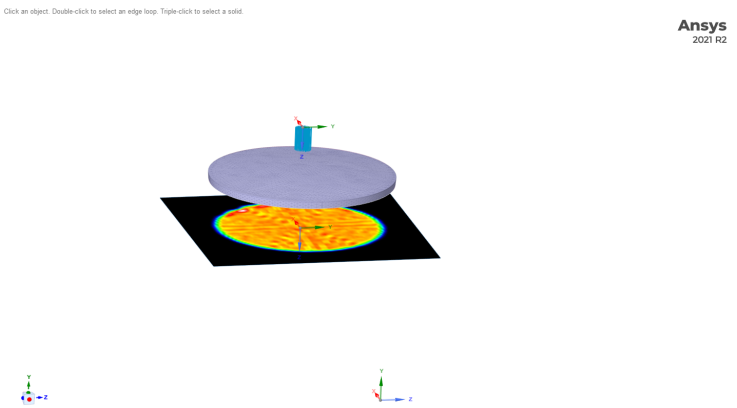

Figure 6: A successful simulation of rays passing through a warped geometry. Note, the rays are actually set to be the diameter of the lens, it is only a rendering setting to depict them smaller.

Scripting in Speos

Fig 7a depicts the script utilized to automate the process of calculating the Zernike Coefficients. Details are given in the Script comments, but a general overview of the process is provided here for the reader's convenience. First the necessary undeformed solid geometry, undeformed mesh, and deformed mesh are imported and placed in the correct positions. They are then run through the warp tool to create an optically suitable surface for the simulation. After this, the warp geometry is added to the direct light simulation (which must be created prior to the script as detailed above). The simulation can then be run to produce an LPF file. With this LPF, an optical surface zernike fit can be output. This is simply a text file that contains the Zernike Coefficients of, in this specific case, the first 7 orders. When placed within a while loop, every deformed timestep can be analyzed.

Radial Distortion Model

Now that the Zernike Coefficients have been acquired for each timestep, we arrive at our final step. Fitting to the camera distortion model:

$$\|\mathbf{r}_{P''}\| = \|\mathbf{r}_{P'}\| \left(1 - c_1 - c_2 - c_3 + c_1 \frac{\|\mathbf{r}_{P'}\|}{R} + c_2 \frac{\|\mathbf{r}_{P'}\|^2}{R^2} + c_3 \frac{\|\mathbf{r}_{P'}\|^3}{R^3}\right)$$

Since a purely radial distortion model is assumed, we must find a way to relate the Zernike Data to a distortion. Rahbar and Faez [3] detail the relation as follows:

$$\frac{\varepsilon_x(x,y)}{f} = \sum_j w_j \frac{\partial Z_j(x,y)}{\partial x},$$
$$\frac{\varepsilon_y(x,y)}{f} = \sum_j w_j \frac{\partial Z_j(x,y)}{\partial y}$$

Where ε_x ε_y are the cartesian displacements of a ray from its ideal impact point (undeformed lens), w is the associated Zernike Coefficient for Zernike Polynomial (Z) of order j. By taking the polar form of the Zernike Polynomials and differentiating with respect to the radius, we acquire a radial distortion model. This process is performed in figure 8a in the appendix. As of now, the script in figure 8a has been tested and calculates a maximum distortion of around 4mm. Though quite high, this is on par with the large focal length of the chosen lens (just over 50mm). Further validation will be necessary to determine accuracy.

Future Steps

Now that the Zernike Coefficients scripts have been created and radial distortion determined, there are several future steps that must be taken. The radial distortion model must still be applied to all deformation timesteps in order to provide long term data for use in the filter. Additionally, the distortion data must still be fit to the third order camera distortion model above. With this, the performance of the filter on real world data may be accurately assessed. These are tasks for the immediate future. Additionally, several refinements should be made to the geometry to reflect realistic satellite design (one such detail being the inclusion of a lens system instead of a single optical surface). Finally, the incident radiation model is assumed to be relatively simple in that exposure over each face is currently set at a uniform value. Using the same ray tracing software as this project (Speos) in conjunction with trajectory and attitude data, a highly accurate thermal model can be created to exactly simulate incident Solar and Earth radiation on a chosen satellite.

References

[1] Thorlabs Product Catalog: Item # LA4246
https://www.thorlabs.com/newgrouppage9.cfm?objectgroup_id=123

[2] Digital Engineering 247 “Free-Floating FEA Models”- Tony Abbey 2015
https://www.digitalengineering247.com/article/free-floating-fea-models/

[3] BLIND CORRECTION OF LENS ABERRATION USING ZERNIKE MOMENTS -Kambiz Rahbar, Karim Faez 2011

Appendix

STL_Export.py : Description

```
1  Model
2  #Run Through List of Deformation Results, Then Export to folder
3  #NOTE: It stops at 200 instead of 201 for this list since the deformation numbering
4  #Is missing 2 (Will not be an issue if recreated with normal numbering)
5
6  for i in range(1,201):
7      result = Model.Analyses[0].Solution.Children[i]
8      result.Activate()
9      STLName="C:\Users\evanw\Desktop\Research\SIOS Lab\STL_Export" + str(i) + ".stl"
10     Graphics.Export3D(STLName, Graphics3DExportFormat.BinarySTL)
11     
12     
13     #"C:\\Users\\evanw\\Desktop\\Summer Research\\SIOS Lab\\ANSYS Files\\Deformed Lens Timesteps\\"
```

Figure 1a: STL_Export.py

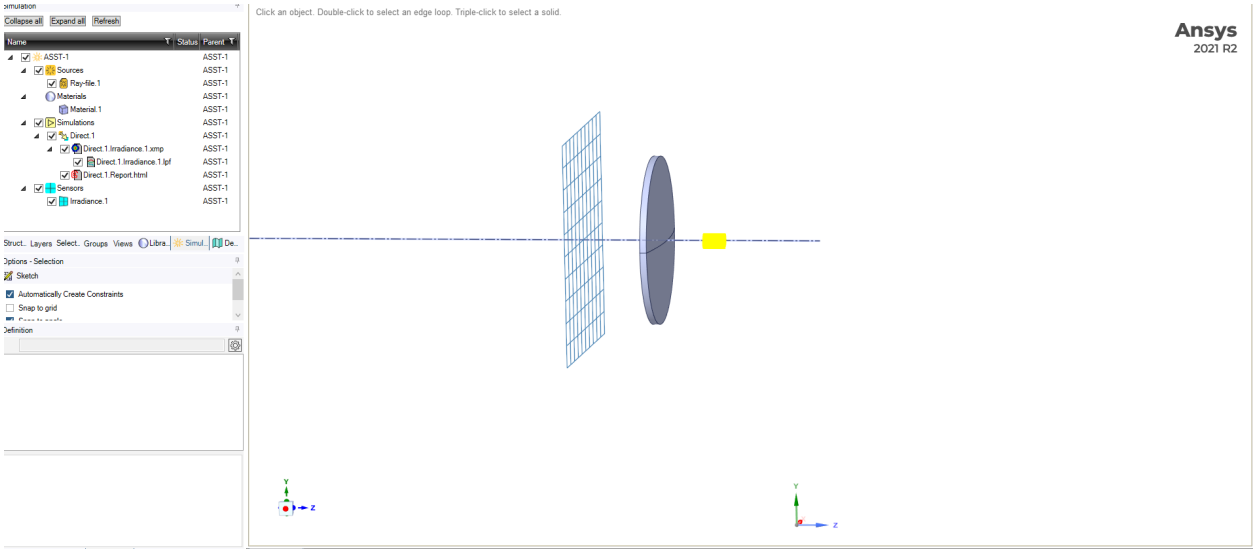

Figure 2a: Speos Setup. The ray file is depicted on the right (in yellow) the lens geometry in the middle, and the irradiance sensor on the left. Note the propagation direction of the light file and the integration direction of the irradiance sensor must be set correctly in order to sense rays. The Geometry need only be set as part of the direct light simulation.

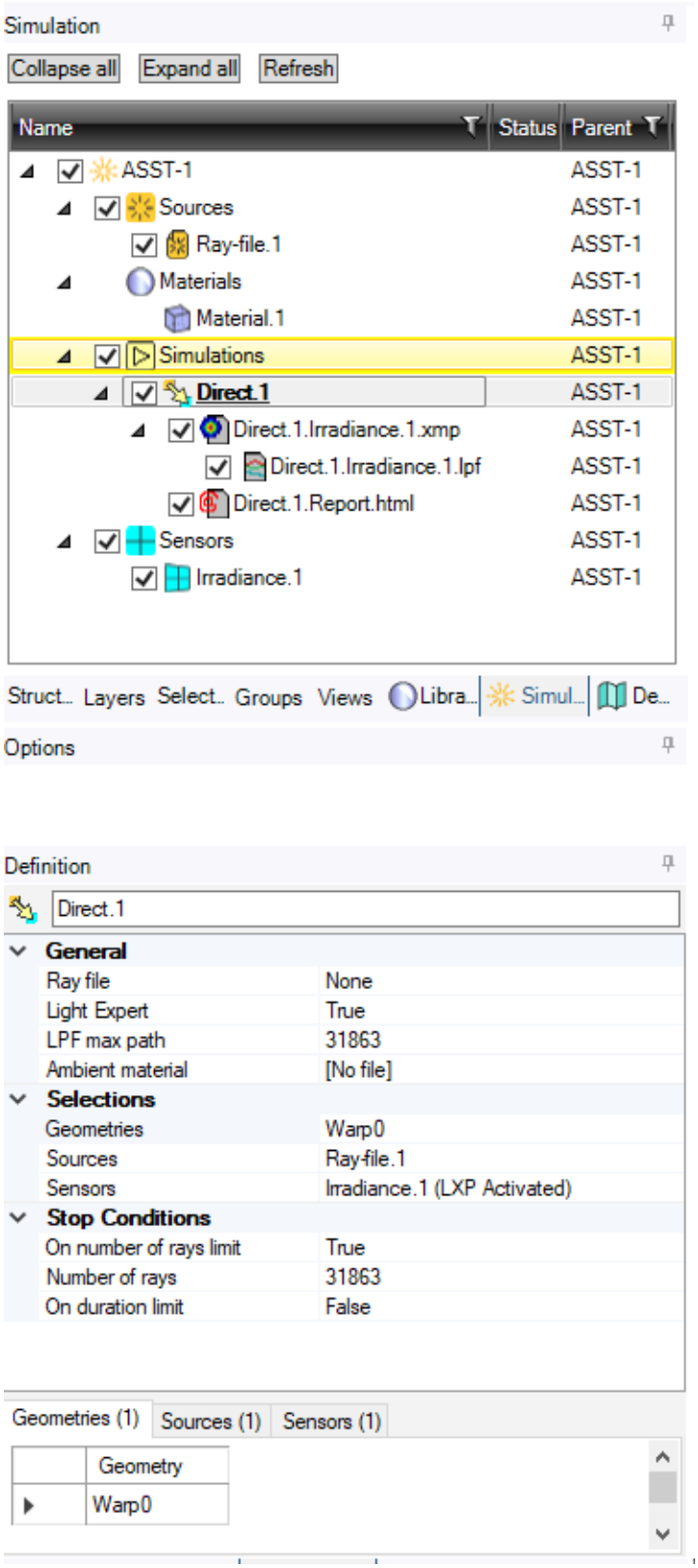

Figure 3a: The simulation tree for Speos. The material and ray file must be created as separate files (both actions can be done through the UI banner)

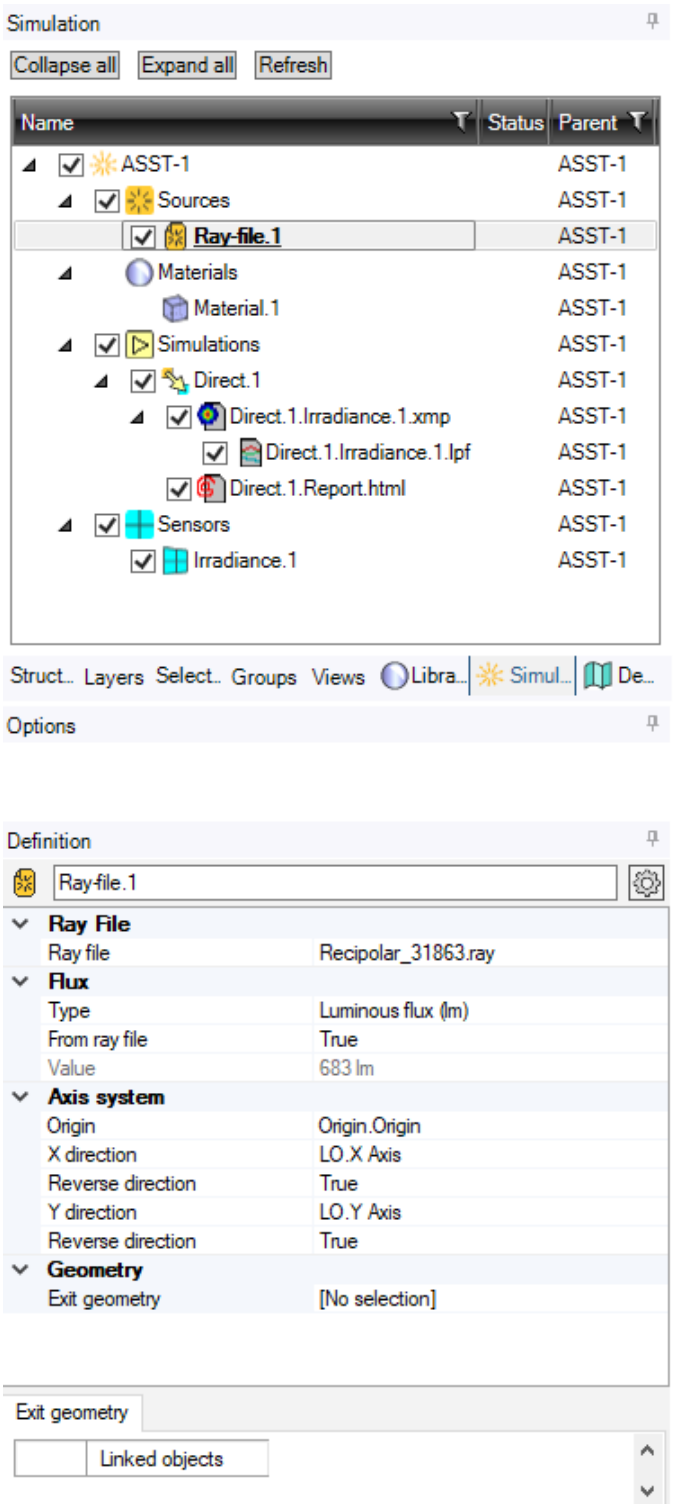

Figure 4a: Ray File Settings

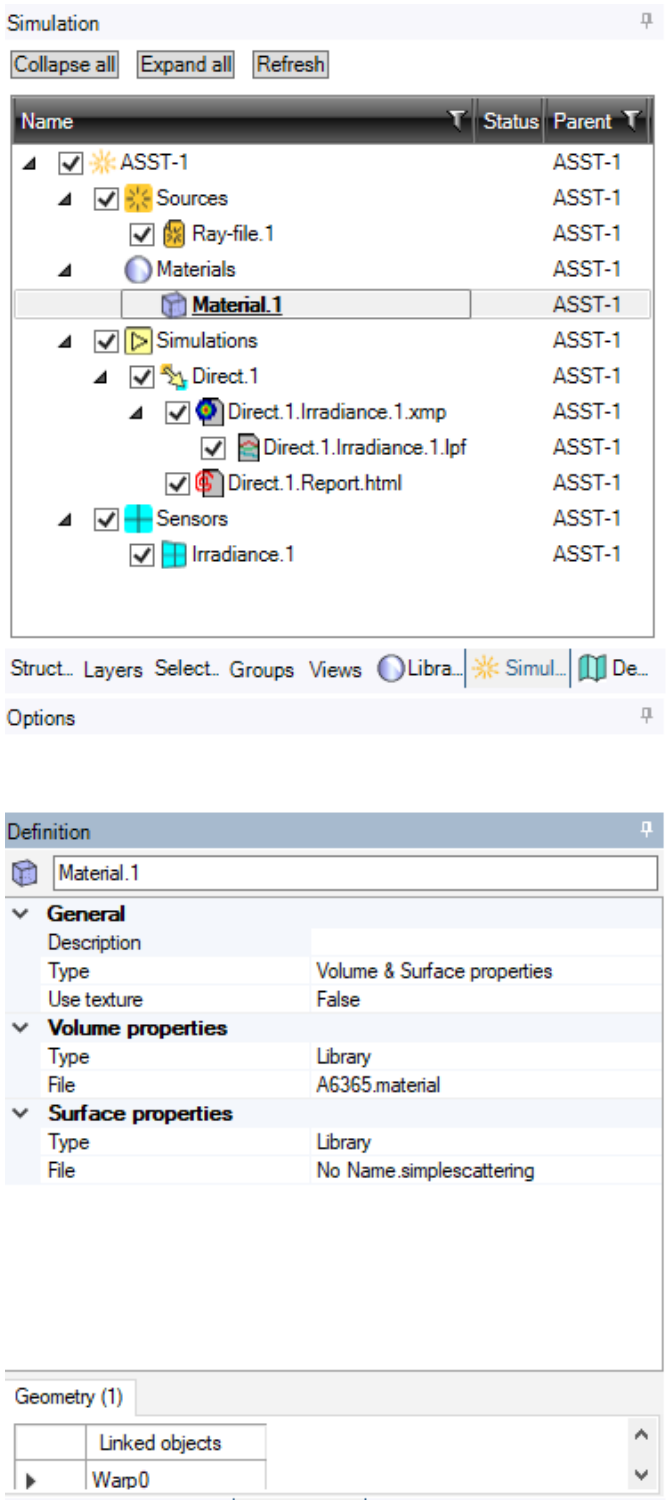

Figure 5a: Material Settings

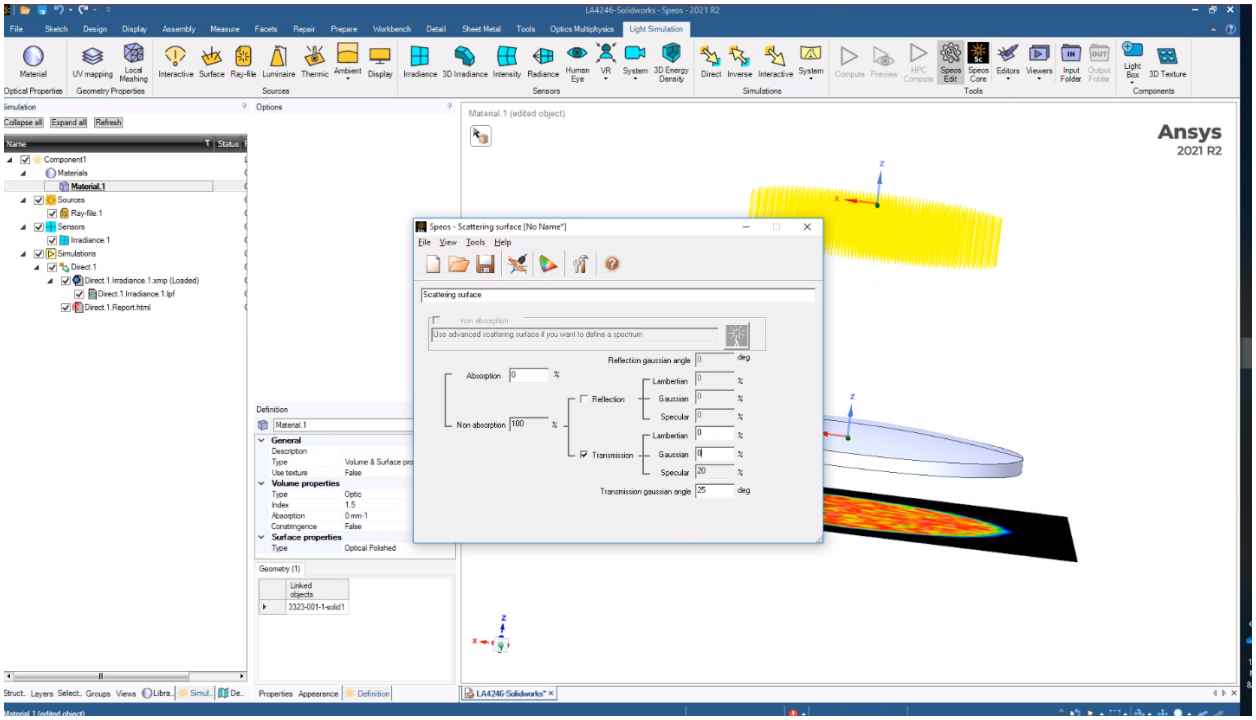

Figure 6a: Custom Material Properties

```python
# Python Script, API Version = V21 Beta
# Python Script, API Version = V21 Beta
import os
import shutil
import subprocess
import popen2

#NOTE: You must set up the irradiance sensor, material, Ray file, and Light Sim manually before proceeding with this automated execution

#Step 1, import relavent meshes under active component. Transform as needed

#Step 2, set names and run warper

#Step 3, find identifier for Warp, export warp to new folder, then delete irrelavent bodies

#Step 4, run light simulations

#Step 5, Call OPD Solver and Calculate Zernike Coefficients

#Step 6, delete and repeat for all deformations




n=1

while n<2:


  DocumentInsert.Execute(r"C:\Users\evanw\Desktop\Research\SIOS Lab\LA4246-Solidworks.STEP", FileSettings5, GetMaps("22233eeb"))

  #Move the undeformed geometry outside the component
  selection = GetRootPart().Components[0].Content.Bodies[0]
  selection=Selection.Create(selection)
  component = GetRootPart()
  result = ComponentHelper.MoveBodiesToComponent(selection, component, False, Info4)

  #Move up to correct origin
  selection = Selection.Create(GetRootPart().Bodies[0])
  upToSelection = Selection.Create(GetRootPart().CoordinateSystems[0])
  anchorPoint = Move.GetAnchorPoint(selection)
  options = MoveOptions()
  result = Move.UpTo(selection, upToSelection, anchorPoint, options, Info5)

  # Delete Empty Component
  selection = Selection.Create(GetRootPart().Components[0])
  result = Delete.Execute(selection)
  # EndBlock

  #Up to this point imports a undeformed geometry, moves it to the correct point, and deletes the empty component
```

```
51
52  #Now for the undeformed mesh
53
54  # Insert From File Undeformed Mesh
55  DocumentInsert.Execute(r"C:\Users\evanw\Desktop\Research\SIOS Lab\Undeformed_SAT.stl", FileSettings6, GetMaps("68a04dcd"))
56  # EndBlock
57
58  # Rotate About X Handle
59  selection = Selection.Create(GetActivePart().Components[0].Content.Meshes[0])
60  axis = Move.GetAxis(selection, HandleAxis.X)
61  options = MoveOptions()
62  result = Move.Rotate(selection, axis, DEG(90), options, Info6)
63  # EndBlock
64
65
66  #Move the undeformed mesh outside the component
67  selection = GetRootPart().Components[0].Content.Meshes[0]
68  selection=Selection.Create(selection)
69  component = GetRootPart()
70  result = ComponentHelper.MoveBodiesToComponent(selection, component, False, Info4)
71
72  # Move Undeformed Mesh Up to Origin
73  selection = Selection.Create(GetRootPart().Meshes[0])
74  upToSelection = Selection.Create(GetRootPart().CoordinateSystems[0])
75  anchorPoint = Move.GetAnchorPoint(selection)
76  options = MoveOptions()
77  result = Move.UpTo(selection, upToSelection, anchorPoint, options, Info5)
78  # EndBlock
79
80  #Delete Old Mesh Component
81  selection = Selection.Create(GetRootPart().Components[0])
82  result = Delete.Execute(selection)
83
84  #Insert Deformed Mesh Rn Just the First one
85  DocumentInsert.Execute(r"C:\Users\evanw\Desktop\Research\SIOS Lab\STL_Export\STL_Export1.stl", FileSettings6, GetMaps("68a04dcd"))
86
87  #Do all the same stuff as the undeformed mesh
88  # Rotate About X Handle
89  selection = Selection.Create(GetActivePart().Components[0].Content.Meshes[0])
90  axis = Move.GetAxis(selection, HandleAxis.X)
91  options = MoveOptions()
92  result = Move.Rotate(selection, axis, DEG(90), options, Info6)
93
94  #Move the deformed mesh outside the component
95  selection = GetRootPart().Components[0].Content.Meshes[0]
96  selection=Selection.Create(selection)
97  component = GetRootPart()
98  result = ComponentHelper.MoveBodiesToComponent(selection, component, False, Info4)
99
100 # Move deformed Mesh Up to Origin
```

```
selection = Selection.Create(GetRootPart().Meshes[1])
upToSelection = Selection.Create(GetRootPart().CoordinateSystems[0])
anchorPoint = Move.GetAnchorPoint(selection)
options = MoveOptions()
result = Move.UpTo(selection, upToSelection, anchorPoint, options, Info5)

#Delete Old Mesh Component
selection = Selection.Create(GetRootPart().Components[0])
result = Delete.Execute(selection)

#TRANSFORMATIONS FINISHED

#Run Warper Command
originalCAD=GetActivePart().Bodies[0]

originalMesh=GetRootPart().Meshes[0]

deformedMesh=GetRootPart().Meshes[1]

SpaceClaim.Api.V21.Scripting.Internal.CustomMethods.WarpBodyByMeshNew(originalCAD, originalMesh, deformedMesh, 0, 1e-4)

#One Note About the Warp command, the precision I am currently running (1e-4) produces a solid body.
#I believe this is the correct process as higher precision produces surface bodies instead. Warrants further
#Investigation

#Delete Non-Warp Objects, we do this to reduce clutter and keep consistant array references
#May be more efficient way to do this, but minimal cost compared to Warp, Optics commands

#Delete Body Component
selection = Selection.Create(GetRootPart().Bodies[0])
result = Delete.Execute(selection)

#Delete Undeformed Mesh Component
selection = Selection.Create(GetRootPart().Meshes[0])
result = Delete.Execute(selection)

#Delete Deformed Mesh Component
selection = Selection.Create(GetRootPart().Meshes[0])
result = Delete.Execute(selection)


#OPTICS SIM
warp=GetActivePart().Bodies[0]

# Add Warp To Simulation
object = SpeosSim.SimulationDirect.Find("Direct.1")
#object = SpeosSim.SimulationDirect(Direct13)
#selection = Body6
object.Geometries.Set(warp)

```

```
# Assign Material to Warp
object = SpeosSim.Material.Find("Material.1")
#object = SpeosSim.Material(Material12)
selection = warp
object.VolumeGeometries.Set(selection)

#Run Simulation

object = SpeosSim.SimulationDirect.Find("Direct.1")
object.Compute()


#Need to export LPF's Now


LPF_Num="\\" + str(n)+".lpf"



#Create Folders
number=r"C:\Users\evanw\Desktop\Research\SIOS Lab\Zernike_Results"+"\\"+str(n)
os.mkdir(number)

ASMB=r"C:\Program Files\ANSYS Inc\v212\scdm\Addins\OpticsMultiphysics\Resources\ASMB_EXE\ASMB_EXE.exe"
LPF=r"C:\Users\evanw\Desktop\Research\SIOS Lab\ANSYS Files\SatP3Gr8Jambory_files\dp0\ASST-1\DM\SPEOS output files\ASST-1\Direct.1.Irradiance.1.lpf"
Output=r"C:\Users\evanw\Desktop\Research\SIOS Lab\Zernike_Results"+"\\"+str(n)

#Execute ASMB Solver From Command Line

subprocess.call([ASMB, "-OPL", "SWF","-f1", LPF, "-outp", Output, "-n","Project_1",  "-OPLsz", "True", "-outc","f", "-outi", "e", "f", "g","h", "-outr", "i" ])

#Delete Old Irradiance Sim and LPF or it wont write new results
```

Figure 7a: Warp Script

```
%Function to import and Process Zernike Results into Radial Distortion
%Model

clear;
clc;


%Read .DAT Zernike Files
p=1;
f=501.8e-3; %current lens from thorlabs focal length

syms r theta

%Encode Zernike Polynomials up to 7th order (only using first 10 right now)
% z{0}=diff(1);
z{1}(r,theta)=diff((r*cos(theta)),r);
z{2}=diff((r*sin(theta)),r);
z{3}=diff((2*r^2-1),r);
z{4}=diff((r^2*cos(2*theta)),r);
z{5}=diff((r^2*sin(2*theta)),r);
z{6}=diff(((3*r^2-2)*r*cos(theta)),r);
z{7}=diff(((3*r^2-2)*r*sin(theta)),r);
z{8}=diff((6*r^4-6*r^3-1),r);
z{9}=diff((r^3*cos(3*theta)));
z{10}=diff((r^3*sin(3*theta)));
z=z';

zz=matlabFunction(cell2sym(z));


while p<5

    path="C:\Users\evanw\Desktop\Research\SIOS Lab\ANSYS Files\A Fresh Start
\Zernike_Results\"+p+"\OSZF\layer-0-Surface-0.DAT";
    %T=readtable("C:\Users\evanw\Desktop\Research\SIOS Lab\ANSYS Files\A Fresh
 Start\Zernike_Results\8\OSZF\layer-0-Surface-0.DAT");
    T=readtable(path);
    %Convert to Array
    A=table2array(T);
    i=0;
    j=1;
    %Now we need to remove first two rows of each file

    while i<2
        i=i+1;
        A(j)=[];
    end

    %Then perform Summation of Zernike Results to calculate ex(x,y), ey(x,y)
    %for a number of sampled points along each radial cooridinate, use to get
    %sample mean deviation from P' along with Covariance
```

```
    distortion=zeros(360,10);

    %Inputting R and theta values intot this sym arrary
    %evaluates their values for every element. Use this for radial distortion


    %Radial Distortion Discretized Across 360 and 10 Radial Sections
    for t=1:360 %discretize each degree
        for d=1:10  %split into 10 radial steps

            zzc=zz(d*7.5e-3,t).*A(2:11); %Zernike Derivatives Multiplied
            %By Coefficients
            distortion(t,d)=f*sum(zzc);  %Disortion for each discretized step
        end
    end
    RESULT(:,:,p)=distortion; %A 3D Array of concatenated results for all time
    p=p+1;
end
```

Figure 8a: Radial Distortion Calculations