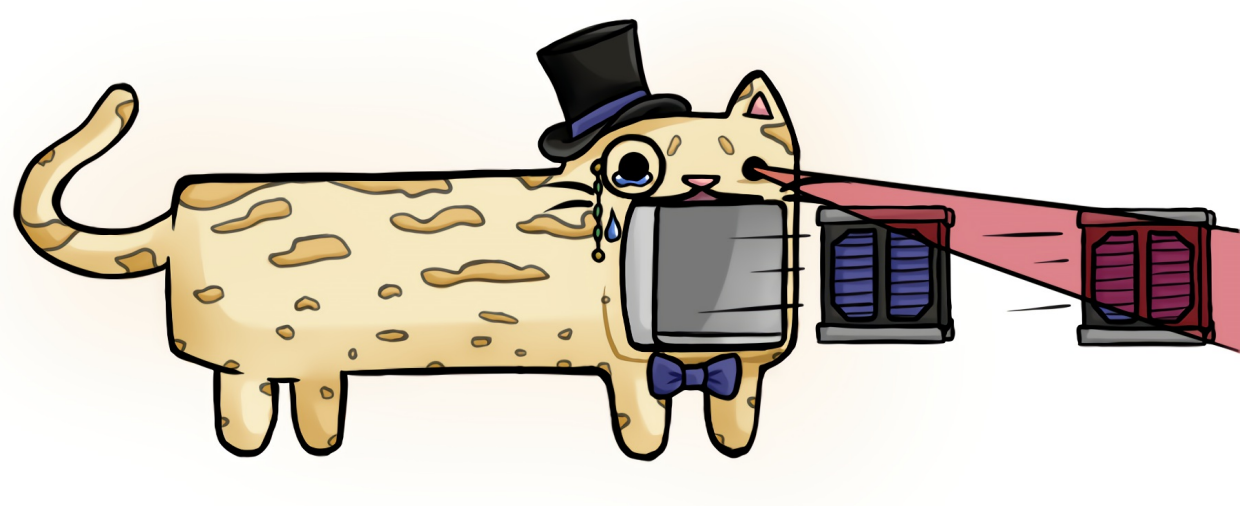
VANTAGE_Simulation

this development documentation was auto-generated from changeset a506a42cd7112284a17da5ef564f8eb108ae46f6

For optimal viewing of this document (and all `.md files), try opening it in a text editor that supports syntax highlighting for markdown *.md` files (e.g. Sublime Text 2+).*



A YAML-powered Python project to interface with C4D / Blensor to simulate the deployment of CubeSats from a NanoRacks ISS deployer.

This repository is the home for the VANTAGE Simulation project, which is a component of the larger VANTAGE (2018-19) project at CU Boulder. You can find more of the software developed for the VANTAGE project on the VANTAGE organization page.

About this Repo

This repo contains the models, documentation, and code necessary to simulate any VANTAGE use-case CubeSat deployment using both monochrome and ToF sensing.

We use Cinema 4D R20 (C4D) to simulate our monochrome camera's properties and Blensor to simulate our ToF flight sensor.

The important directories contained in this repo are:

- `config` : this dir contains the YAML configuration files used to completely automate the creation of a C4D animation case.
- `3d_assets` : this dir contains the 3D deployer model as well as the 3D cubesat models used in the simulation.
- `docs` : the homeland for wild simulation documentation

Directory Structure

The following directory structure shows the directory structure if you run the full simulation framework with a configuration file named: `configName = SPS_template.yaml`. The `runC4D.py` code will automatically build a "case" directory from this config file, named with the following convention `configName + '-' + '%H_%M_%S_%Y_%m_%d'`.

```
.
├── **4_Simulation_Cases**
│   ├── ...
│   └── *SPS_template-09_13_17_2019_04_19*
│       ├── *ToF_Data*
│       │   ├── Sample1
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00001.pcd
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00002.pcd
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00003.pcd
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00004.pcd
│       │   │    └── ...
│       │   ├── Sample2
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00001.pcd
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00002.pcd
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00003.pcd
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00004.pcd
│       │   │    └── ...
│       │   ├── Sample3
│       │   │    └── ...
│       │   ├── Sample4
│       │   │    └── ...
│       │   └── ...
│       ├── *Optical_Data*
```

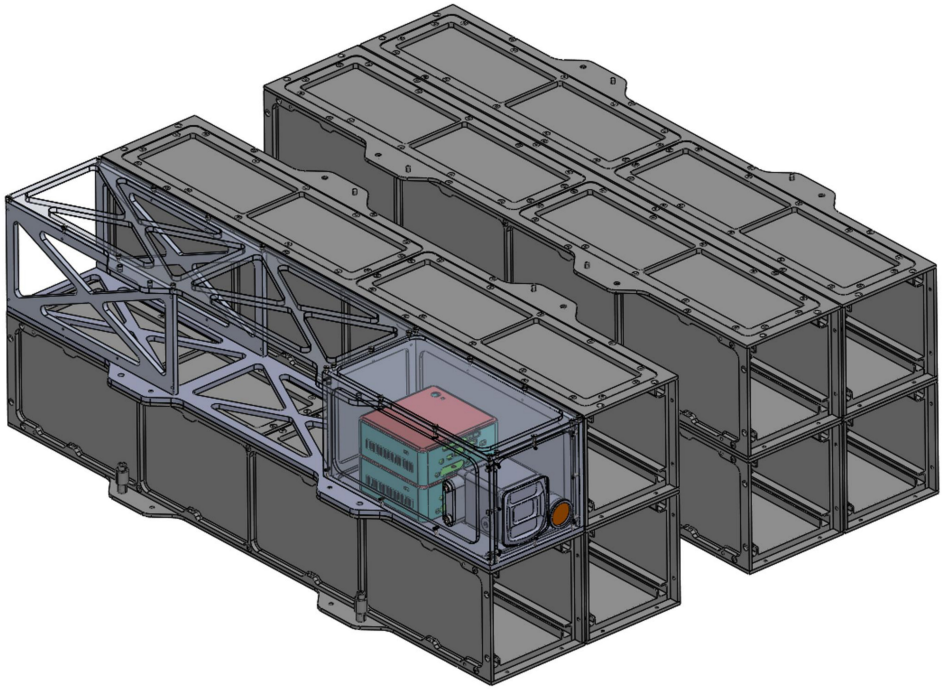
```
│       │   ├── SPS_template-09_13_17_2019_04_19_Camera_a0000.png
│       │   ├── SPS_template-09_13_17_2019_04_19_Camera_a0001.png
│       │   ├── SPS_template-09_13_17_2019_04_19_Camera_a0002.png
│       │   └── ...
│       ├── SPS_template-09_13_17_2019_04_19.blend
│       ├── SPS_template-09_13_17_2019_04_19.c4d
│       ├── SPS_template-09_13_17_2019_04_19.fbx
│       ├── SPS_template_truth_data.json
│       └── SPS_template.yaml
└── **VANTAGE_Simulation**
    ├── 3d_assets
    │   ├── 1U_Slug.SLDPRT
    │   ├── 2U_Slug.SLDPRT
    │   ├── 3U_Slug.SLDPRT
    │   ├── 4U_Slug.SLDPRT
    │   ├── 5U_Slug.SLDPRT
    │   ├── 6U_Slug.SLDPRT
    │   └── NR_Dual_Quadpack_Simulation_Assem.SLDPRT
    ├── cleanCaseToFDirs.py
    ├── *config*
    │   ├── configArchive
    │   │   ├── config_simulation_Josh_CDR.yaml
    │   │   └── ...
    │   ├── final_vantage_sim_configs
    │   │   ├── VTube4_DTube3_CubeSats3U2U1U_Speed100cmps.yaml
    │   │   └── ...
    │   ├── config_simulation_template.yaml
    │   └── **SPS_template.yaml**
    ├── currSimConfigFile
    ├── docs
    │   ├── deployer_VANTAGE_geometry_and_coordinate_frame_defs.pdf
    │   ├── VANTAGE_tube4_cubesat_tube3_63cm_downrange.png
    │   └── ...
    ├── README.md
    ├── runBlensor.py
    └── runC4D.py
```

Simulation Parameterization / Geometry Definition

The definition of deployer variables used in the parameterization and case creation can be found in (shown below):

`<VANTAGE_SIMULATION>\docs\deployer_VANTAGE_geometry_and_coordinate_frame_defs.pdf`

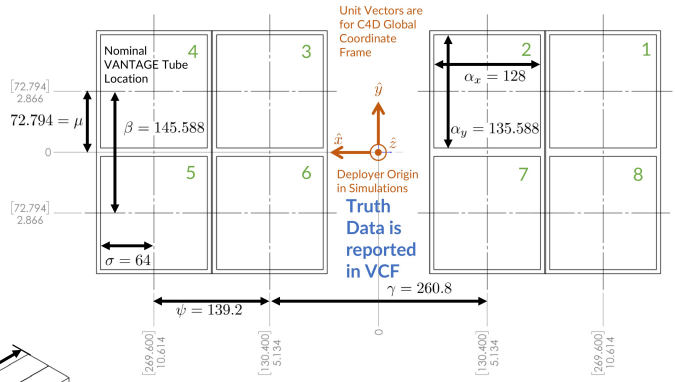
The `docs` directory contains a lot of the relevant information used in the process of creating and validating the simulation.

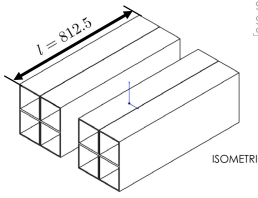


NOTES:
This part is for virtual simulations only

Deployer Geometry Setup

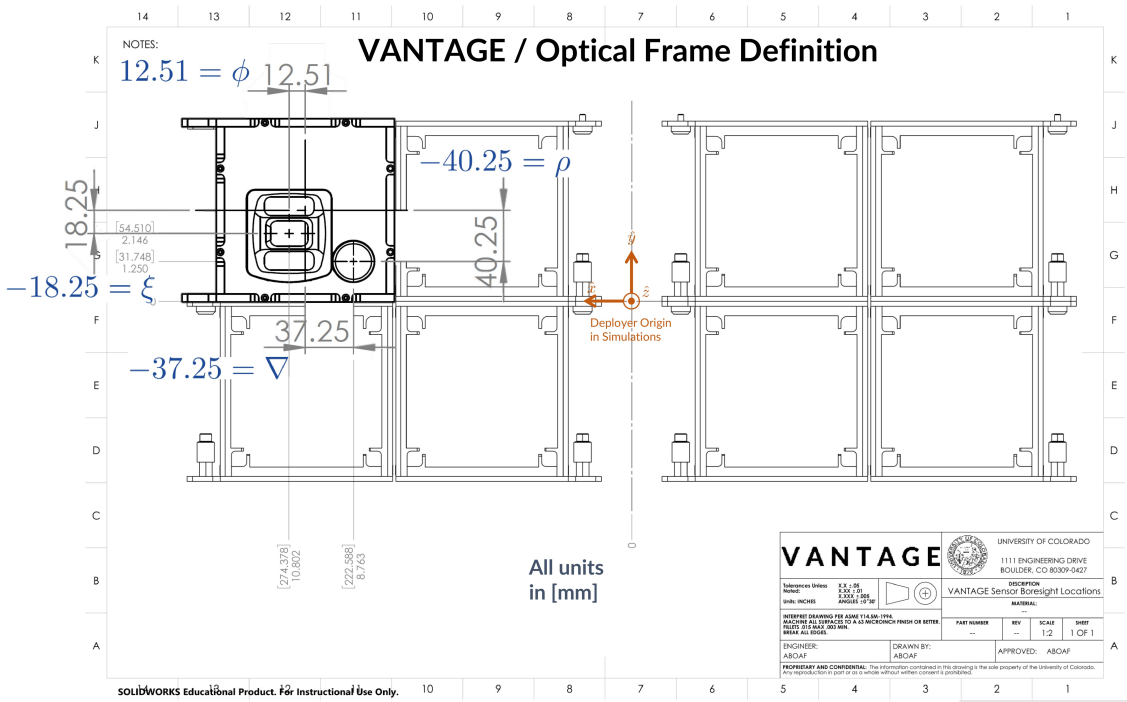
This page is used for understanding how VANTAGE's and CubeSats' locations are automagically setup in the C4D script.





All units in [mm]

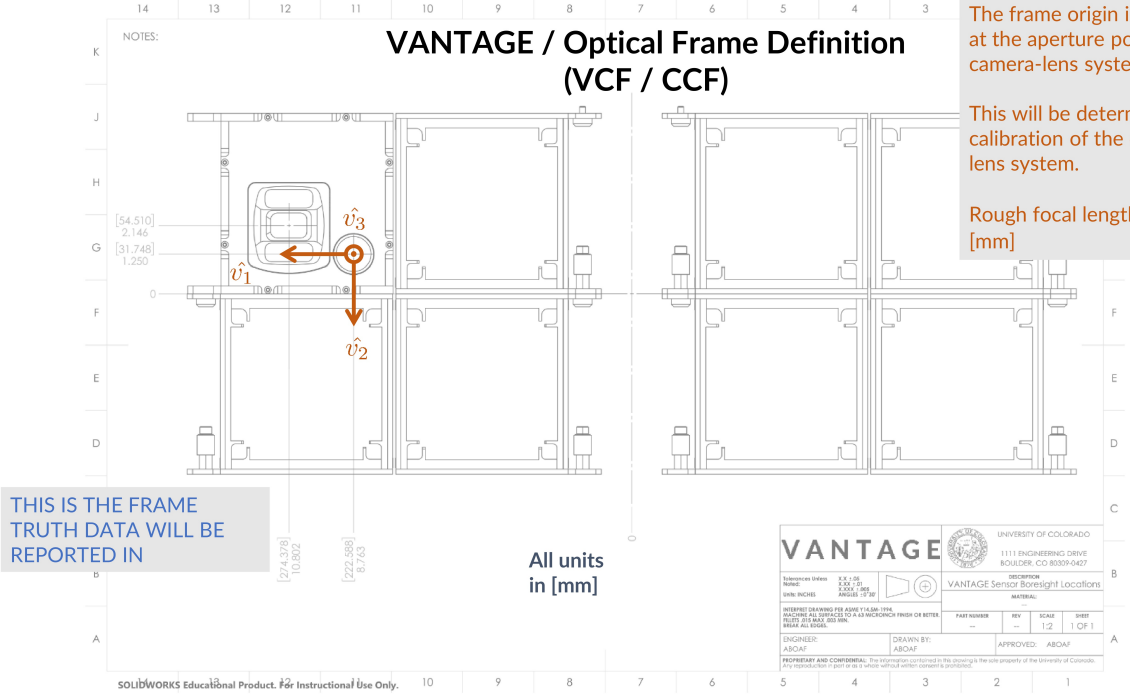
VANTAGE
UNIVERSITY OF COLORADO
1111 ENGINEERING DRIVE
BOULDER, CO 80309-0427
DESCRIPTION: NanoRacks Deployer Simulation Assembly
MATERIAL:
PART NUMBER: ----
REV: A
SCALE: 1:3
SHEET: 1 OF 1
ENGINEER: Aboof
DRAWN BY: Aboof
APPROVED: Aboof



The frame origin is located at the aperture point of the camera-lens system.

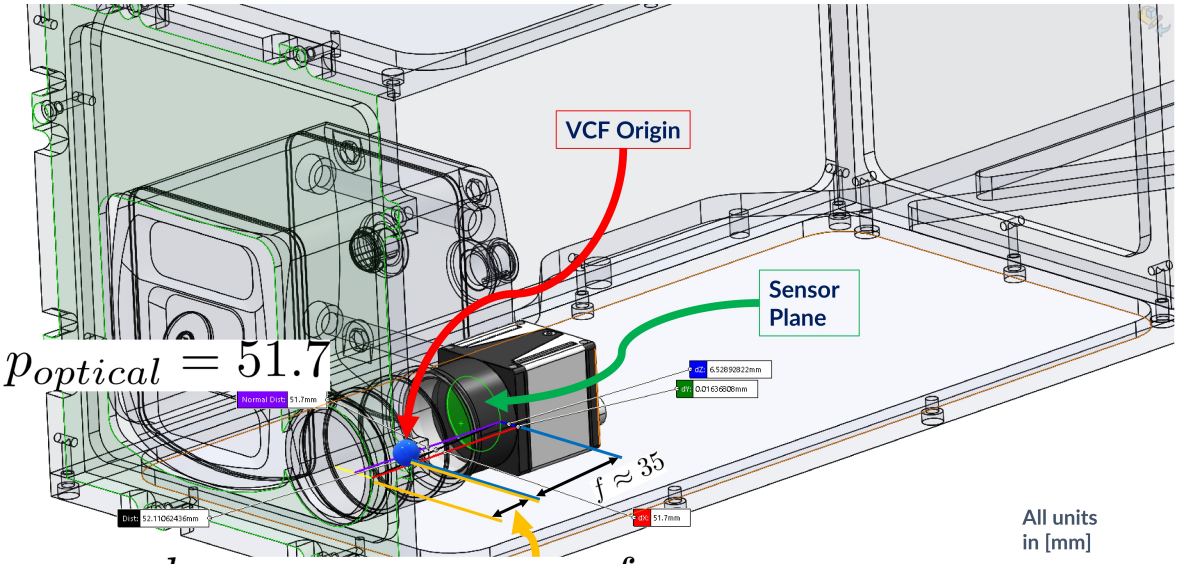
This will be determined by calibration of the camera-lens system.

Rough focal length, f, is 35 [mm]



THIS IS THE FRAME TRUTH DATA WILL BE REPORTED IN

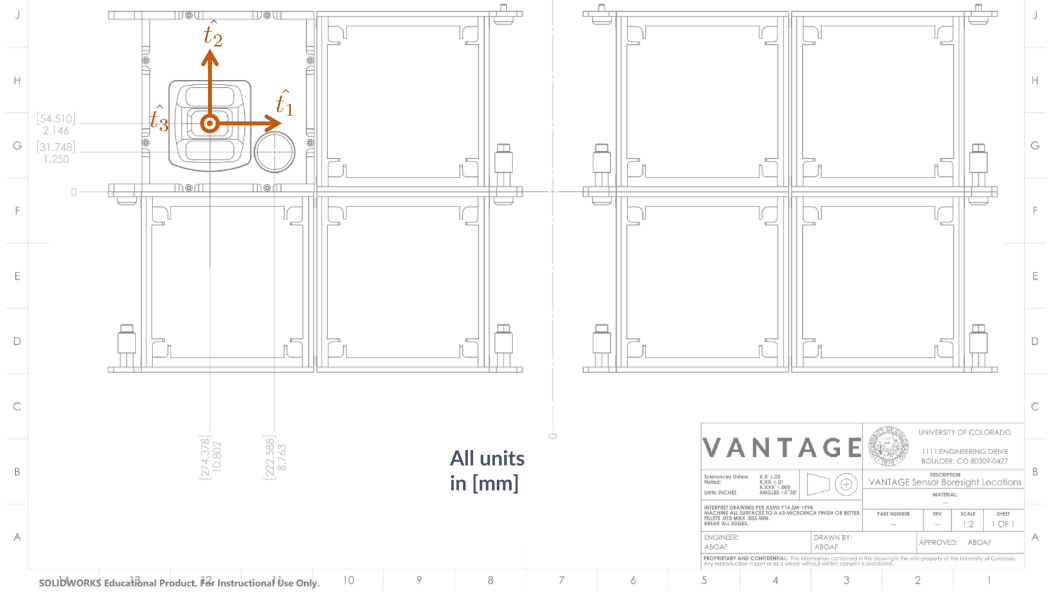
Location of Optical Sensor Plane and VCF Origin Relative to VANTAGE Front Plate



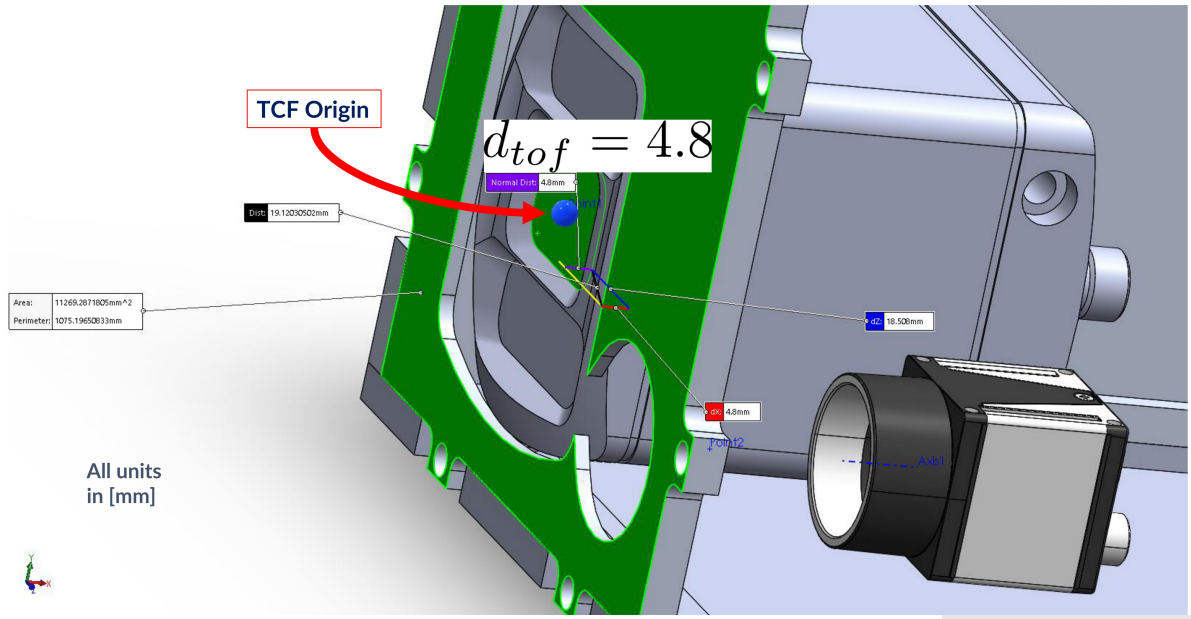
$$d_{optical} = p_{optical} - f$$

The frame origin is located on the glass of the ToF lens.

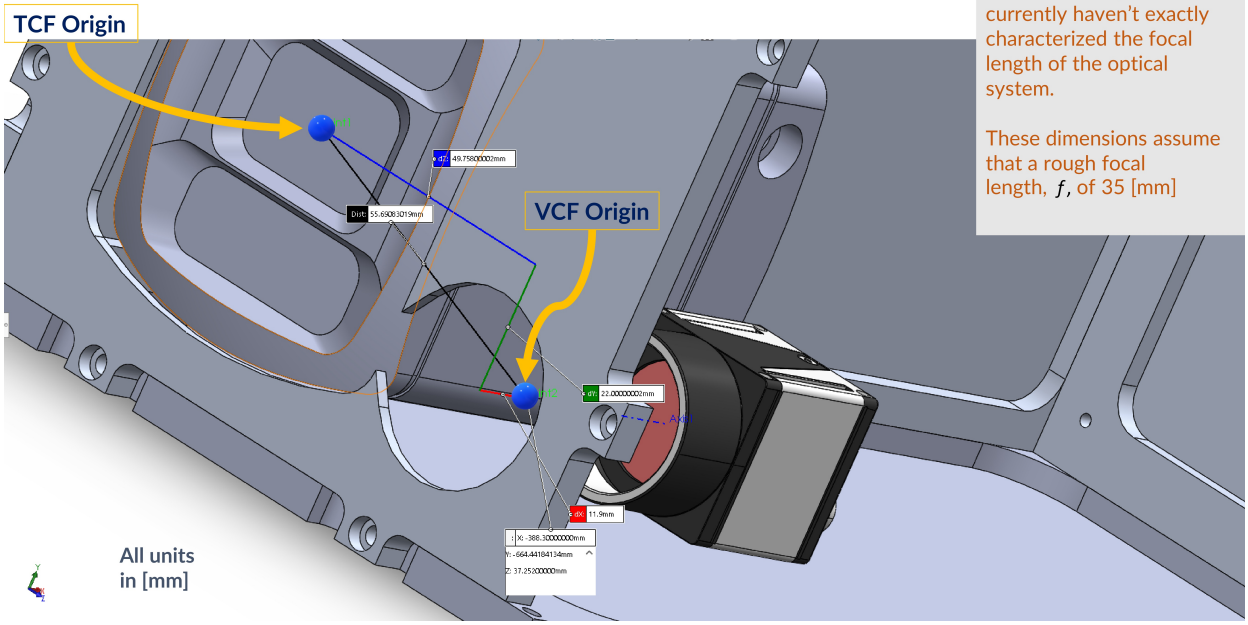
ToF Frame Definition (TCF)



Location of TCF Origin Relative to VANTAGE Front Plate

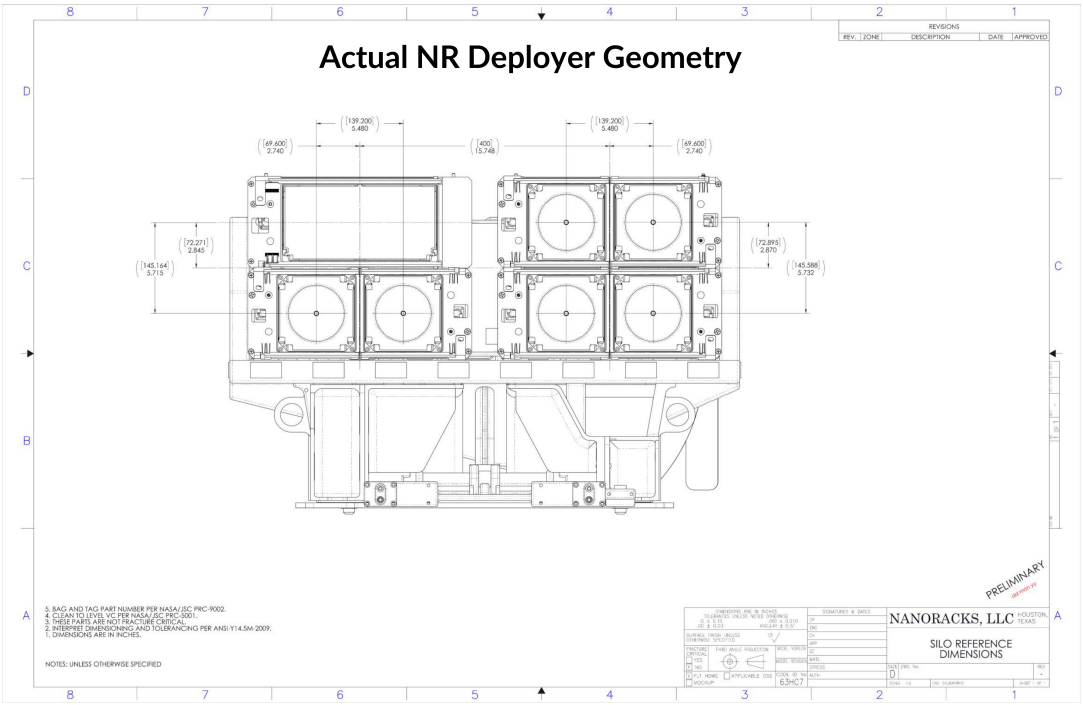


TCF to VCF Distance



The exact dimensions of this definition are rough in the x direction, as we currently haven't exactly characterized the focal length of the optical system.

These dimensions assume that a rough focal length, f, of 35 [mm]



Description of Simulation Parameters

We have heavily scripted the creation of a C4D simulation case. Currently, the entire simulation is automated with the following being some major *settings you can change*:

- which deployment tube you are launching out of
- where the global coordinate system is centered - originally set to the center of tube 1 (11/5)
- the sequence and size of CubeSats that will be deployed
- The Optical / ToF camera parameters
- the motion of each individual CubeSat (linear and rotational velocities of each CubeSat centroid)

WARNING: we do not currently check for collision, so make sure you don't prescribe motion that will cause the CubeSats collide

The description of what each parameter is should be evident in the location you set it, whether it be in the YAML config, or in the source itself.

How to Set the Simulation Parameters

I would highly recommend using a text editor which supports YAML syntax highlighting - it will make editing the config file much less error prone and easier to read.

In order to change these settings, make a copy of: `<VANTAGE_SIMULATION_LOCATION>/config/config_simulation_template.yaml` in the `<VANTAGE_SIMULATION_LOCATION>/config/` directory.

USE `config_simulation_template.yaml` as your starter config NOT `SPS_template.yaml`
(I'm sorry I accidentally built this whole example with the `SPS_template` for some dumb reason and I'm scared to change it)

Rename your new config file with a meaningful name, following a reasonable config naming convention.

Edit the config file variables as you see fit to create the correct C4D case. Make sure to visually inspect the case before you run a lengthy simulation.

As I ran out of time, some of the ToF camera simulation parameters must be adjusted in runBlensor.py itself. I am so, so sorry.

Simulation Output Directory -- `4_Simulation_Cases`

Each time you run a simulation case in C4D and then in Blensor (the methodology for doing so is shown in How to Run a C4D Simulation and How to Render with C4D and How to Run a Blensor Simulation, an output "Case Directory" (an example case directory is shown throughout this document as `./4_Simulation_Cases/SPS_template-09_13_17_2019_04_19`) is created corresponding to the name and current time of the configuration file specified in `currSimConfigFile` (see How to Run a C4D Simulation for more details on configuration).

This directory will contain the following files, as seen in the "Directory Structure" section of this document:

```
.
├── 4_Simulation_Cases
│   ├── ...
│   └── SPS_template-09_13_17_2019_04_19
│       ├── ToF_Data
│       │   ├── Sample1
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00001.pcd
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00002.pcd
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00003.pcd
│       │   │    ├── Sample1_SPS_template-09_13_17_2019_04_19_noisy00004.pcd
│       │   │    └── ...
│       │   ├── Sample2
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00001.pcd
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00002.pcd
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00003.pcd
│       │   │    ├── Sample2_SPS_template-09_13_17_2019_04_19_noisy00004.pcd
│       │   │    └── ...
```

```
|       |   ├── Sample3
|       |   |    └── ...
|       |   ├── Sample4
|       |   |    └── ...
|       |   └── ...
|       ├── Optical_Data
|       |   ├── SPS_template-09_13_17_2019_04_19_Camera_a0000.png
|       |   ├── SPS_template-09_13_17_2019_04_19_Camera_a0001.png
|       |   ├── SPS_template-09_13_17_2019_04_19_Camera_a0002.png
|       |   └── ...
|       ├── SPS_template-09_13_17_2019_04_19.blend
|       ├── SPS_template-09_13_17_2019_04_19.c4d
|       ├── SPS_template-09_13_17_2019_04_19.fbx
|       ├── SPS_template_truth_data.json
|       └── SPS_template.yaml
```

For the next description, I will generalize the specific string associated with the example case directory name - `SPS_template-09_13_17_2019_04_19`) - as `<CASE_DIR_NAME>` (e.g. `SPS_template-09_13_17_2019_04_19` <==> `<CASE_DIR_NAME>`). `<CASE_DIR_NAME>` refers to *any and all* case directory names generated during the simulation process.

`<CASE_DIR_NAME_LITE>` refers to the case directory name without the date string appended to it (e.g. `<CASE_DIR_NAME> = 'SPS_template-09_13_17_2019_04_19'` , `<CASE_DIR_NAME_LITE> = 'SPS_template'`)

A detailed description of the major case directory (`<CASE_DIR_NAME>`) components are as follows:

- `./4_Simulation_Cases/<CASE_DIR_NAME>/ToF_Data` : This directory contains pointcloud (`*.pcd`) data from the Blensor Simulation. Each "Sample" directory contains a full set of pointclouds generated from simulated noisy ToF scans over a simulated CubeSat deployment. The reason for having multiple scans is to have multiple full scans of the same deployment, each with different random noise, to test the robustness of our pointcloud processing algorithms against sensor noise. *This directory is only populated if you manually elect to produce ToF pointcloud in Blensor -- see How to Run a Blensor Simulation for details*

- `./4_Simulation_Cases/<CASE_DIR_NAME>/Optical_Data` : This directory contains ray-traced renders (`*.png`) from the simulated VANTAGE monochrome camera generated by a C4D render. These simulated images have been calibrated and tested to be representative of images taken by the real monochrome camera we employ (see `*/docs/side_by_side_comparison.png`) *This directory is only populated if you manually elect to produce rendered images with C4D -- see How to Render with C4D for details*

- `./4_Simulation_Cases/<CASE_DIR_NAME>/<CASE_DIR_NAME>.blend` : This is the blender file containing the full deployment animation and the full ToF simulation state after it has run. This is an optional, but recommended file to save so you can go back later and see exactly what ToF parameters were used. *see How to Run a Blensor Simulationfor details*

- `./4_Simulation_Cases/<CASE_DIR_NAME>/<CASE_DIR_NAME>.c4d` : This is the C4D file containing the C4D scenario with all objects, cameras, and C4D render settings. Saving this file is suggested to the user after the C4D simulation creation process. *see How to Run a C4D Simulation for details*

- `./4_Simulation_Cases/<CASE_DIR_NAME>/<CASE_DIR_NAME>.fbx` : This filetype (`.fbx`) is a generic 3D file format that will contain all 3D objects, their animations, and their parameters (including the camera parameters). This file is created in C4D and then imported into Blensor to load in the same deployment scenario. Saving this file is suggested to the user after the C4D simulation creation process. *see How to Run a C4D Simulation and How to Run a Blensor Simulation*

- `./4_Simulation_Cases/<CASE_DIR_NAME>/<CASE_DIR_NAME_LITE>_truth_data.json` : This file, in the JSON data serialization format (please, please dear god use something like this (e.g. YAML, JSON, XML, etc.) *for all* file I/O) contains all of the actual 3D positions of each CubeSat's centroid in VCF ([cm]) at each time step ([s]). The JSON schema for the truth data is *roughly* this (below generated for `./4_Simulation_Cases/SPS_template-09_13_17_2019_04_19/SPS_template_truth_data.json`):

```
"$schema": "TruthData",
# the entire truth data is an array of time values and the centroid locations
# of all CubeSats at each time value
"type": "array",
"items": [
  {
    # time step object, only contains an actual number
    "type": "object",
    "properties": {
      # the simulation time step value
      "t": {
        "type": "number"
      },
      # the CubeSats State object
      "pos": {
        "type": "object",
        "properties": {
          # unique name given to each CubeSat based on the file used to
          # create the object and its deployment order
          "launch_num_3__3U_Slug.SLDPRT": {
            "type": "array",
            "items": [
              {
                # v_hat_1 centroid location component in VCF [cm]
                "type": "number"
              },
              {
                # v_hat_2 centroid location component in VCF [cm]
                "type": "number"
              },
              {
```

```
            # v_hat_3 centroid location component in VCF [cm]
            "type": "number"
          }
        ]
      },
      # unique name given to each CubeSat based on the file used to
      # create the object and its deployment order
      "launch_num_0__1U_Slug.SLDPRT": {
        "type": "array",
        "items": [
          {
            # v_hat_1 centroid location component in VCF [cm]
            "type": "number"
          },
          {
            # v_hat_2 centroid location component in VCF [cm]
            "type": "number"
          },
          {
            # v_hat_3 centroid location component in VCF [cm]
            "type": "number"
          }
        ]
      },
      # unique name given to each CubeSat based on the file used to
      # create the object and its deployment order
      "launch_num_1__1U_Slug.SLDPRT": {
        "type": "array",
        "items": [
          {
            # v_hat_1 centroid location component in VCF [cm]
            "type": "number"
          },
          {
            # v_hat_2 centroid location component in VCF [cm]
            "type": "number"
          },
          {
            # v_hat_3 centroid location component in VCF [cm]
            "type": "number"
          }
        ]
      },
      # unique name given to each CubeSat based on the file used to
      # create the object and its deployment order
      "launch_num_2__1U_Slug.SLDPRT": {
        "type": "array",
        "items": [
          {
            # v_hat_1 centroid location component in VCF [cm]
            "type": "number"
          },
          {
            # v_hat_2 centroid location component in VCF [cm]
            "type": "number"
          },
```

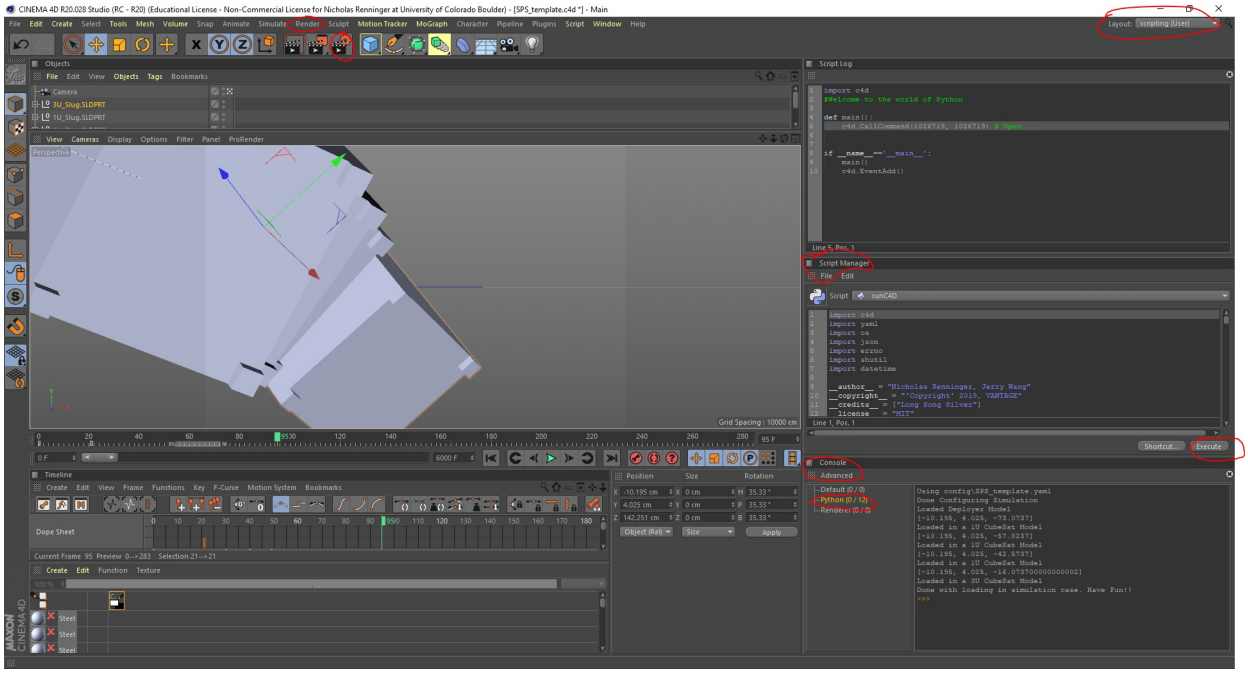
```
                {
                  # v_hat_3 centroid location component in VCF [cm]
                  "type": "number"
                }
              ]
            }
          },
          # a CubeSat centroid array is req. for each CubeSat ALWAYS
          "required": [
            "launch_num_3__3U_Slug.SLDPRT",
            "launch_num_0__1U_Slug.SLDPRT",
            "launch_num_1__1U_Slug.SLDPRT",
            "launch_num_2__1U_Slug.SLDPRT"
          ]
        }
      },
      # must ALWAYS contain both timing and position for each time step
      "required": [
        "t",
        "pos"
      ]
    },

    # repeated t and pos object for each time step simulated
    ...
]
```

- `./4_Simulation_Cases/<CASE_DIR_NAME>/<CASE_DIR_NAME_LITE>.yaml`: This file contains the YAML configuration file that you specify in `./<VANTAGE_SIMULATION_LOCATION>/currSimConfigFile` (see How to Run a C4D Simulation for more details), which controls *almost* all of the simulation parameters (see How to Run a Blensor Simulation on what parameters must be set outside the YAML config file). It is automatically copied to `<CASE_DIR_NAME>` during the C4D simulation setup so you know *exactly* what settings were used to create each simulation (for audibility).

How to Run a C4D Simulation

`runC4D.py`



1) Open C4D (you can get an educational license from the Maxon website - see `./<VANTAGE_SIMULATION_LOCATION>/docs/0 HowToGetCinema4d` for more detailed info)

2) Open the script manager (Script -> Script Manager) and the console (Script -> Console)

3) From the script manager, open `./<VANTAGE_SIMULATION_LOCATION>/runC4D.py`

4) Edit `./<VANTAGE_SIMULATION_LOCATION>/currSimConfigFile` with the name of the config file you would like to run the simulation with (e.g. `SPS_template` to use `SPS_template.yaml` as your input config file).

5) Click the **Execute** button in the C4D Script Manager

6) The simulation will load everything in and set all necessary settings.

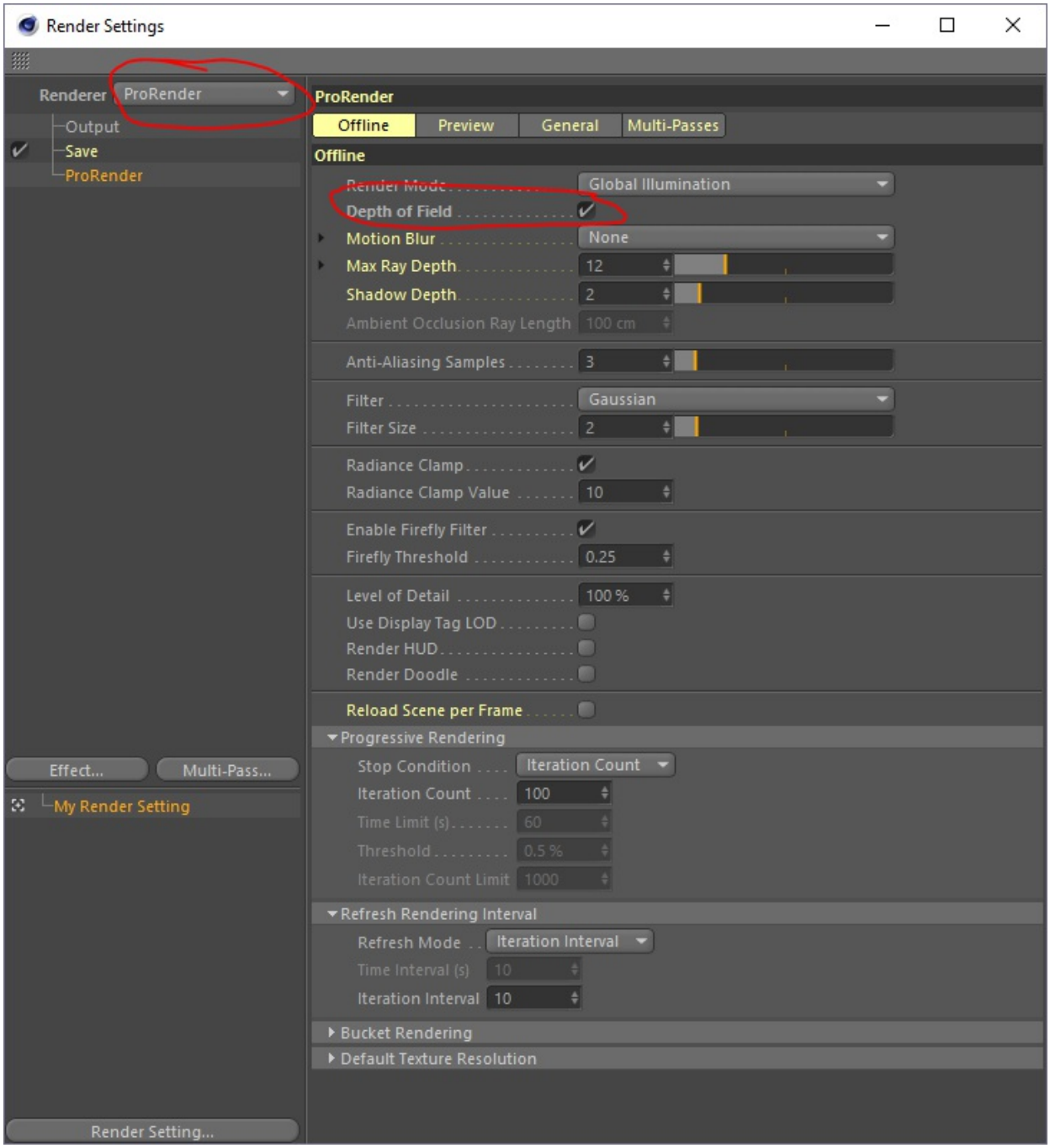
7) A window will pop up asking you to save the c4d file. Navigate to the "case" directory (see the directory structure section for more details) created for the config file you typed in `currSimConfigFile`.

8) At this point you get to name the .c4d file. **Recommended file name:** Copy the name of the case directory (e.g. `SPS_template-09_13_17_2019_04_19`), paste this name into the text field, then append `.c4d` to this name (e.g. `SPS_template-09_13_17_2019_04_19.c4d`).

9) Now you will be asked to name the FBX output (animation / camera data) of the simulation. It is again recommended to use the case dir name + `.fbx` as the fbx output filename (e.g. `SPS_template-09_13_17_2019_04_19.fbx`).

10) If you would like to actually generate representative images from the camera, you will need to **render** the camera's perspective of the deployment. See the How to Render with C4D section for details.

How to Render with C4D

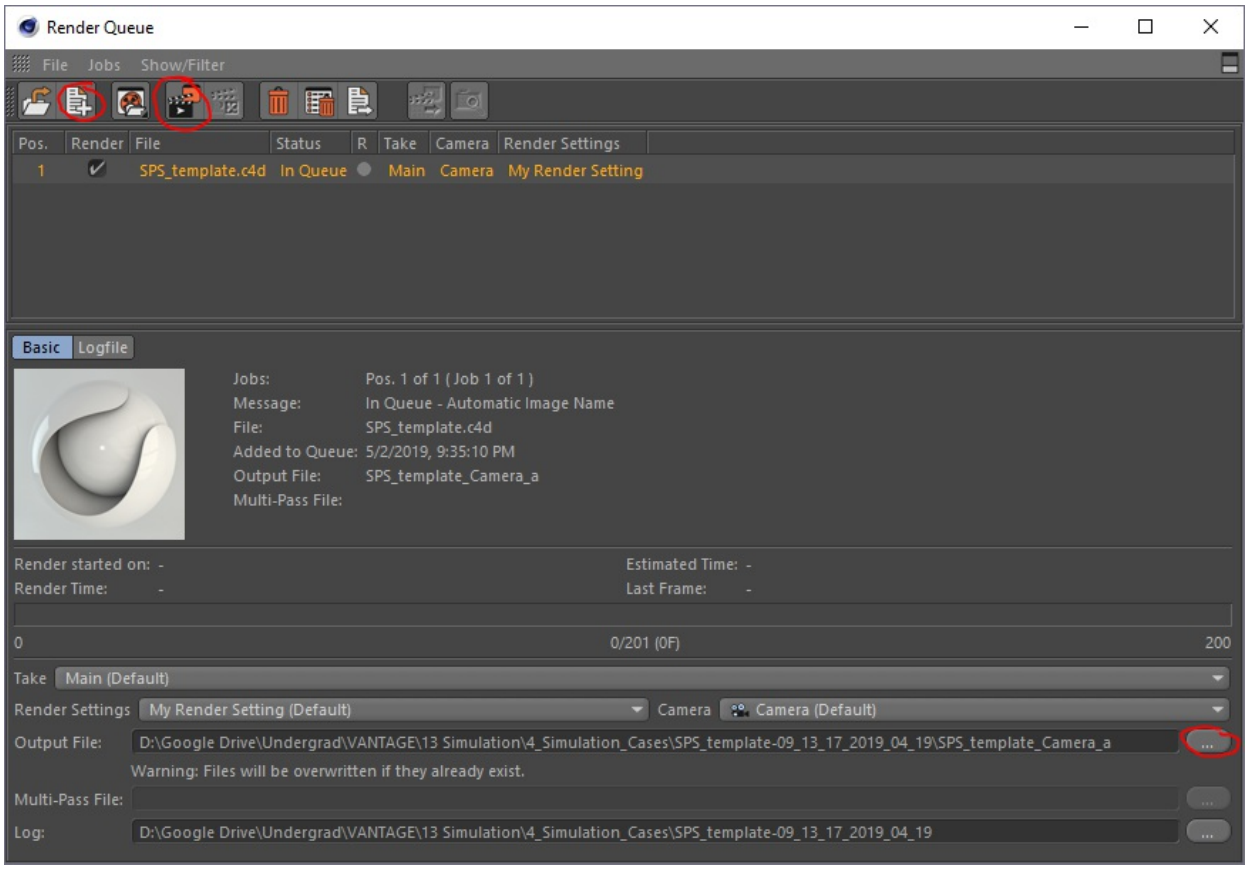


1) After you have completed all steps in How to Run a C4D Simulation, you must now edit the render settings. Go to Render -> Edit Render Settings...

2) In the Render settings, make sure to select the **ProRender** setting. You may have to toggle between render engines before ProRender activates. You must use the ProRender engine to properly render everything. In R20.028, there is a bug that prevents automated choice of render engine. MAKE SURE that the ProRender menu is properly set!

3) In the ProRender settings, make sure you turn ON **Depth of Field**. All other render settings should be automatically set by the python module. Close the render settings menu.

4) To start a queue of renders, go to Render -> Render Queue.



5) "Open" the `.c4d` file in the case directory you would like to render (e.g. `SPS_template-09_13_17_2019_04_19.c4d`)

6) Click the three dots on the right of the "output file" text field to select the output image file location and naming convention.

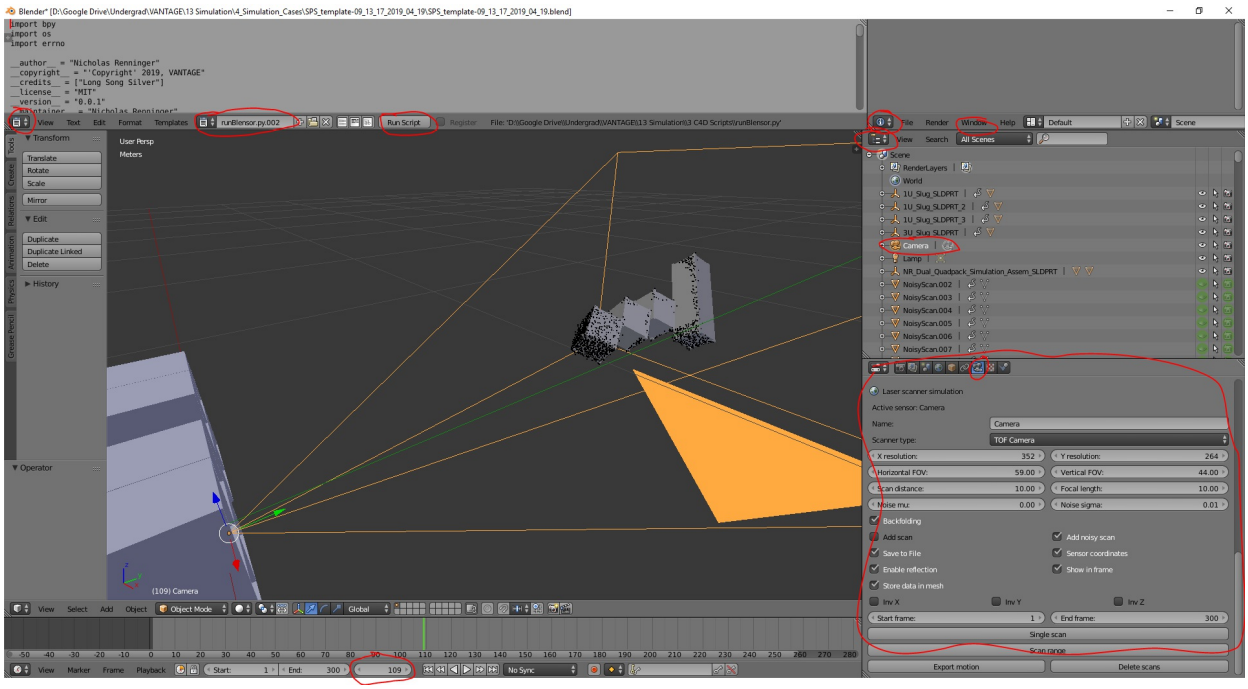
7) Navigate to the case directory for the `.c4d` file you are rendering, then navigate to the `./4_Simulation_Cases/<CASE_DIR_NAME>/Optical_Data/` directory (e.g. `./4_Simulation_Cases/SPS_template-09_13_17_2019_04_19/Optical_Data/`).

8) Copy the case directory name (e.g. `SPS_template-09_13_17_2019_04_19`) into the "File Name" field.

9) Click "Start Render" and watch your CPU and GPU melt for an hour. :)

How to Run a Blensor Simulation

`runBlensor.py`



To begin this section, **you must have completed all steps in How to Run a C4D Simulation first.** C4D is used for the simulation animation building, so you must have the .fbx file output to begin the Blensor simulation process.

1) Start by opening `runBlensor.py` in your favorite text editor -- I prefer using desktop sticky notes.

2) Next, open up Blensor. If you have access to the VANTAGE drive, you will need my modified version of Blensor from the simulation directory, or some of this won't work. :(

3) File -> Import -> .fbx

4) Navigate to the desired simulation case directory and select the .fbx you created in How to Run a C4D Simulation (e.g. `./4_Simulation_Cases/SPS_template-09_13_17_2019_04_19/SPS_template-09_13_17_2019_04_19.fbx`) and *double* click the file to import it.

5) Watch a YouTube video while it loads in.

6) In the "Data Outliner" sub-menu, select *ONLY* the "Camera" Object.

7) Determine how many frames it takes in the simulation for the closest CubeSat to reach its desired distance away from the deployer.

8) Enter this number as `MAX_FRAMES` in `runBlensor.py`. This will make the simulation take a ToF scan every frame from frame 1 to frame `MAX_FRAMES`.

9) Next, in `runBlensor.py`, set `NUM_SIMS_TO_RUN`. This will run the whole stochastic simulation `NUM_SIMS_TO_RUN` times. See "Directory Structure" for how each "Sample" of the sensor simulation over the range frame is stored in the case directory. You will end up with

`NUM_SIMS_TO_RUN` independent sets of noisy sensor data over the scan range defined in the `MAX_FRAMES` setting step.

10) Now, set the `outputCase` variable in `runBlensor.py` to be the name of the case directory you loaded in as an .fbx previously (e.g. `outputCase = 'SPS_template-09_13_17_2019_04_19'`).

11) Now, go back to Blensor. Open the region editor "Text Editor" in Blensor, and open `./<VANTAGE_SIMULATION_LOCATION>/runBlensor.py`.

12) Again, in the "Data Outliner" region editor, ensure you have selected *ONLY* the "Camera" Object.

13) In a different region, open the Scripting "Info" editor. Once this is open, you'll want to go to Window -> Toggle System Console.

14) **NOW you're ready to run!** yay. In the text Editor region editor, click "Run Script".

15) Watch gigabytes of data slowly appear in your case directory ToF_Data directory

16) The only way to tell when the simulation is fully done is to watch the System Console output you just opened, as Blensor uses a threaded dispatch system without any locking implementation. Once the System Console no longer reports output, you will need to clear up the output directories.

USE THE FOLLOWING AT YOUR OWN RISK *The following can be done more safely manually if you don't trust my file deletion regex or the absolute paths supplied.*

These steps should only be done once you are happy with your simulations and are ready to begin using the data for testing VANTAGE software.

17) Open `*/<VANTAGE_SIMULATION_LOCATION>/cleanCaseToFDirs.py` in your favorite sticky note based text editor. Edit `outputDir` to be the correct absolute path to the `4_Simulation_Cases` directory.

18) Navigate to `*/<VANTAGE_SIMULATION_LOCATION>/` from powershell / terminal and then run `*/<VANTAGE_SIMULATION_LOCATION>/cleanCaseToFDirs.py`. This removes all of the non-noisy ToF `.pcd` files from **ALL** `./4_Simulation_Cases/<ANY_CASE_NAME>/ToF_Data` directories.

19) You will have to manually remove more files manually from each "Sample" directory in ToF_Data:

- `./4_Simulation_Cases/<CURRENT_CASE_NAME>/ToF_Data/Sample<XXX>/Sample<XXX>_<CURRENT_CASE_NAME>`
- `./4_Simulation_Cases/<CURRENT_CASE_NAME>/ToF_Data/Sample<XXX>/Sample<XXX>_<CURRENT_CASE_NAME>.pcd`

20) Save the current scence as a `.blend` file in the case directory (e.g.

`./4_Simulation_Cases/<CURRENT_CASE_NAME>/<CURRENT_CASE_NAME>.blend`)

21) Congrats on simulating a ToF Camera!

Contact Info

Questions or feature requests should be directed to:


```
Nicholas Renninger
nicholas.renninger@colorado.edu
```