PhytoOracle

Release 1.0b

Oct 22, 2021

Home

PhytoOracle is a scalable, modular data pipeline for phenomics research. It uses CCTools' Makeflow and WorkQueue distributed task management frameworks. As a result, PhytoOracle significantly reduces data processing times. The pipeline also enables researchers to swap between supported extractors or to easily integrate new extractors. PhytoOracle is, therefore, able to handle the increasing rate of data collection while making it accessible to a wide range of users.

CHAPTER 1

Welcome to PhytoOracle!

PhytoOracle is a scalable, distributed workflow manager for analyzing highthroughput phenotyping data. It is designed to process data from the UA Gantry, but can be adapted to work on data coming from other platforms. PhytoOracle uses a master-worker framework for distributed computing (HPC, Cloud, etc.) and can run jobs on nearly all unix-like environments. Access our Github here.

1.1 Supported Sensors & Pipelines

Sensor	Sensor Name	Data Description
StereoTopRGB	Prosilica GT3300C	Stereo RGB images. Identifies plants; measuring plant area
FlirIr	FLIR A615, 45°	Infrared images. Measures temperature of plants
PSII	LemnaTec custom based of an Allied Vision Manta camera	Fluorescence images. Measures chlorophyll fluorescence for calculating plant photosynthetic potential.
Scanner3DTop	Custom 3D Fraunhofer	Laser scanning images. Generates a point cloud for measuring physical structure of plants.
Hyperspectral (VNIR/SWIR)	Custom Headwall Photonics	Hyperspectral images. Collects and processes information from across the electromagnetic spectrum for a wide variety of phenotypes (e.g., vegetation indices)

1.2 Pipeline Structure

All of the pipelines follow the same structure that allows for accessiblility, scalability, and modularity. The steps are:

1. Setting up the Master interactive node and Worker nodes on the HPC
2. Cloning the pipeline of choice
3. Staging the data
4. Editing the scripts
5. Launching the pipeline

1.3 Acknowledgements

This project partially built on code initially developed by the TERRA-REF project and Ag-Pipeline team. We thank the University of Arizona Advanced Cyberinfrastrcture Concept class of 2019 for additional work. Logo credit: Christian Gonzalez.

1.4 Issues and Questions

If you have questions, raise an issue on the GitHub page.

For specific workflows and adapting a pipeline for your own work contact:

- Emmanuel Gonzalez: emmanuelgonzalez [at] email.arizona.edu
- Michele Cosi: cosi [at] email.arizona.edu

For plant detection and plant clustering:

- Travis Simmons: travis.simmons [at] ccga.edu

For the orthomosaicing algorithm:

- Ariyan Zarei: ariyanzarei [at] email.arizona.edu

CHAPTER 2

Using PhytoOracle on the HPC

2.1 Overview

This guide will walk you through the necessary steps required to launch PhytoOracle's pipelines onto a High Performance Computer system using interactive nodes (as tested on the University of Arizona's HPC running the PBS Pro and SLURM RJMS - Resource and Job Management System).

The interactive node functions as the "foreman" within the time-saving "foreman-worker" framework. The interactive node distributes the computational load, connecting to the "workers" through its IP address (or assigned job name) and job management scripts from the PhytoOracle repository.

2.2 Software Requirements

Ensure that your HPC is running CentOS 7 and has these software installed:

- Python 3 (tested with python v 3.8)
- Singularity (tested with singularity v 3.5.3)
- CCTools (tested with CCTools v 7.0.19)
- iRODS (tested with iRODS v 4.2.7)
- Git (tested with Git v 1.7.1)

2.3 Launching Interactive Node

To launch an interactive node:

```
qsub -I -N phytooracle -W group_list=<your_group_list> -q <priority> -l␣
↪select=1:ncpus=<CPU_N>:mem=<RAM_N>gb:np100s=1:os7=True -l walltime=<max_hour_N>:0:0
```

or

```
srun --nodes=1 --mem=<RAM_N> --ntasks=1 --cpus-per-task=<CPU_N> --time=<max_hour_N> --
↪job-name=po_mcomp --account=<your_group_list> --partition=<priority> --mpi=pmi2 --
↪pty bash -i
```

replace <your_group_list>, <priority>, <CPU_N>, <RAM_N>, <max_hour_N> with your preferred settings.

An example on the UA HPC is:

```
qsub -I -N phytooracle -W group_list=lyons_lab -q standard -l␣
↪select=1:ncpus=28:mem=224gb:np100s=1:os7=True -l walltime=12:0:0
```

or

```
srun --nodes=1 --mem=470GB --ntasks=1 --cpus-per-task=94 --time=24:00:00 --job-
↪name=po_mcomp --account=lyons-lab --partition=standard --mpi=pmi2 --pty bash -i
```

Once the interactive node is running, clone the PhytoOracle repository:

```
git clone https://github.com/uacic/PhytoOracle
```

Before proceeding, note the IP address of the interactive node. You can find the IP address with `ifconfig`. This will be used for connecting the manager to the workers.

`cd` (change directory) into the desired pipeline and continue.

2.4 Launching Workers

Create an executable script according to your HPC's RJMS system. If using PBS Pro, use your preferred editor to create a `.pbs` script, if using SLURM create a `.sh` using the following templates:

PBS Pro:

```
#!/bin/bash
#PBS -W group_list=<your_group_list>
#PBS -q <priority>
#PBS -l select=<N_nodes>:ncpus=<CPU_N>:mem=<RAM_N>gb
#PBS -l place=pack:shared
#PBS -l walltime=<max_hour_N>:00:00
#PBS -l cput=<max_compute_N>:00:00
module load singularity

export CCTOOLS_HOME=/home/<u_num>/<username>/cctools-<version>
export PATH=${CCTOOLS_HOME}/bin:$PATH

/home/<U_ID>/<USERNAME>/cctools-<version>/bin/resource_monitor -O log-flirIr-makeflow␣
↪-i 2 -- work_queue_factory -T local <INTERACTIVE_NODE_ADDRESS>.<HPC_SYSTEM> 9123 -w␣
↪12 -W 16 --workers-per-cycle 10 --cores=1 -t 900
```

SLURM:

```
#!/bin/bash
#SBATCH --account=lyons-lab --partition=standard
#SBATCH --job-name="phytooracle"
```

(continues on next page)

(continued from previous page)

```
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=94
#SBATCH --time=24:00:00
#module load singularity
module load python/3.8

export CCTOOLS_HOME=/home/<u_num>/<username>/cctools-<version>
export PATH=${CCTOOLS_HOME}/bin:$PATH

/home/<U_ID>/<USERNAME>/cctools-<version>/bin/work_queue_worker -M PhytoOracle_FLIR -
↪t 900
```

As before, change the highlighted `<fields>` to preferred settings.

Here are examples on the UA HPC system, using "u1" as the user number and "hpcuser" as the username, looks like:

PBS Pro:

```
#!/bin/bash
#PBS -q standard
#PBS -l select=1:ncpus=28:mem=224gb:np100s=1:os7=True
#PBS -W group_list=lyons-lab
#PBS -l place=pack:shared
#PBS -l walltime=5:00:00
#PBS -l cput=140:00:00
#module load unsupported
#module load ferng/glibc
module load singularity

export CCTOOLS_HOME=/home/u1/hpcuser/cctools-7.1.5-x86_64-centos7
export PATH=${CCTOOLS_HOME}/bin:$PATH
cd /home/u1/hpcuser/data_output_folder

/home/u1/hpcuser/cctools-7.1.5-x86_64-centos7/bin/work_queue_factory -T local
↪<commander_IP_address>.ocelote.hpc.arizona.edu 9123 -w 24 -W 26 --workers-per-cycle␣
↪10 --cores=1 -t 900
```

It is important to note that lines 12, 14, and 27 will have to be personalized, and the commander IP address must be specified in line 27.

SLURM:

```
#!/bin/bash
#SBATCH --account=windfall --partition=windfall
#SBATCH --job-name="phytooracle"
#SBATCH --nodes=1
#SBATCH --ntasks=1
#SBATCH --ntasks-per-node=94
#SBATCH --time=24:00:00
#module load singularity
module load python/3.8

export CCTOOLS_HOME=/home/u12/cosi/cctools-7.1.6-x86_64-centos7
export PATH=${CCTOOLS_HOME}/bin:$PATH

/home/u1/hpcuser/cctools-7.1.6-x86_64-centos7/bin/work_queue_worker -M PhytoOracle_
↪FLIR -t 900
```

Save your changes and submit with:

PBS Pro:

```
qsub <filename>.pbs
```

SLURM:

```
sbatch <filename.pbs>
```

Depending on the traffic to the HPC system, this may take some time. You can search for your submitted job using:

PBS Pro:

```
qstat -u username
```

SLURM:

```
squeue -u username
```

The HPC setup is now complete. Navigate to the pipeline of your choice to continue:

- StereoTopRGB
- FlirIr
- PSII
- Stereo3DTop
- Hyperspectral

CHAPTER 3

Obtaining Pipeline Related Containers

All the code used throughout the pipeline is containerized through Docker and hosted on DockerHub. We use Singularity to execute containers on the HPC system.

Each container is first downloaded and stored in `.simg` format to maximise time efficiency. We suggest creating a folder containing all containers in `.simg` format close to your root directory and adding the path to the folder to the `process_one_set.sh` pipeline scripts.

To create a `.simg` file you will require Singularity to be installed and executable, then do:

```
singularity build <name_of_container>.simg docker://<dockeruser>/<container>:<version>
```

For Example:

```
singularity build stitch_plots.simg docker://phytooracle/stitch_plots:latest
```

3.1 Full list of containers

3.1.1 StereoTopRGB

Container	DockerHub Repo	GitHub Link
bin2tif	`docker://phytooracle/rgb_bin_to_tif:latest`	https://github.com/phytooracle/rgb_bin_to_tif
collect_gps	`docker://phytooracle/rgb_flir_collect_gps:latest`	https://github.com/phytooracle/rgb_flir_collect_gps
Orthomosaicing	`docker://ariyanzarei/full_geocorrection:latest`	https://github.com/ariyanzri/Lettuce_Image_Stitching
replace_gps	`docker://phytooracle/rgb_flir_edit_gps:latest`	https://github.com/phytooracle/rgb_flir_edit_gps
plotclip	`docker://phytooracle/rgb_flir_plot_clip_geojson:latest`	https://github.com/phytooracle/rgb_flir_plot_clip_geojson
Plant detection	`docker://phytooracle/rgb_flir_plant_detection`	https://github.com/phytooracle/rgb_flir_plant_detection
Plant clustering	`docker://phytooracle/rgb_flir_plant_clustering:latest`	https://github.com/phytooracle/rgb_flir_plant_clustering

3.1.2 FlirIr

Container	DockerHub Repo	GitHub Link
flir2tif	`docker://phytooracle/flir_bin_to_tif_s10:latest`	https://github.com/phytooracle/flir_bin_to_tif_s10
collect_gps	`docker://phytooracle/rgb_flir_collect_gps:latest`	https://github.com/phytooracle/rgb_flir_collect_gps
Orthomosaicing	*docker://ariyanzarei/full_geocorrection:latest*	https://github.com/ariyanzri/Lettuce_Image_Stitching
replace_gps	`docker://phytooracle/rgb_flir_edit_gps:latest`	https://github.com/phytooracle/rgb_flir_edit_gps
flir_field_stitch	`docker://phytooracle/flir_field_stitch:latest`	https://github.com/phytooracle/flir_field_stitch
plotclip	`docker://phytooracle/rgb_flir_plot_clip_geojson:latest`	https://github.com/phytooracle/rgb_flir_plot_clip_geojson
flir_meantemp	`docker://phytooracle/flir_meantemp:latest`	https://github.com/phytooracle/flir_meantemp

3.1.3 PSII

Container	DockerHub Repo	GitHub Link
cleanmetadata	`docker://AgPipeline/moving-transformer-cleanmetadata:latest`	https://github.com/AgPipeline/moving-transformer-cleanmetadata
bin2tif	`docker://phytooracle/psii_bin_to_tif:latest`	https://github.com/phytooracle/psii_bin_to_tif
resizetif	`docker://phytooracle/psii_resize_tif:latest`	https://github.com/phytooracle/psii_resize_tif
flir_field_stitch	`docker://phytooracle/flir_field_stitch:latest`	https://github.com/phytooracle/flir_field_stitch
plotclip	`docker://phytooracle/rgb_flir_plot_clip_geojson:latest`	https://github.com/phytooracle/rgb_flir_plot_clip_geojson
psii_segmentation	`docker://phytooracle/psii_segmentation:latest`	https://github.com/phytooracle/psii_segmentation
psii_fluorescence_aggregation	`docker://phytooracle/psii_fluorescence_aggregation:latest`	https://github.com/phytooracle/psii_fluorescence_aggregation

3.1.4 Scanner3DTop

Container	DockerHub Repo	GitHub Link
3D Merge-Ply	`docker://phytooracle/3d_merge_ply:latest`	https://github.com/phytooracle/3d_merge_ply

CHAPTER 4

Running the StereoTopRGB Pipeline for Detecting Plant Area Data

This pipeline extracts plant area data from image files. This guide provides demo data you can use follow along with and ensure the pipeline is functional. Before starting, change to `alpha` branch with `git checkout alpha`.

4.1 Pipeline Overview

StereoTopRGB currently uses 7 different programs for the analytical pipeline:

Program	Function	Input	Output
bin2tif	Converts bin compressed files to geotiff	`image.bin`, `metadata.json`	`image.tif`
collect_gps	Collects GPS coordinates from all geotiff files	`image.tif`	`collected_coordinates.csv`
Orthomosaicing	Finds best possible coordinates of all geotiffs	`collected_coordinates.csv`	`corrected_coordinates.csv`
replace_gps	Applies corrected GPS coordinates to images	`corrected_coordinates.csv`, `image.tif`	`corrected_image.tif`
plotclip	Clips geotiffs to the plot	`corrected_image.tif`, `shapefile.geojson`	`plot.tif`
Plant detection	Detects plants over days	`plot.tif`	:`genotype.csv`
Plant clustering	Tracks plants over days	`genotype.csv`	:`pointmatching.csv`

4.2 Running the Pipeline

Note: At this point, we assume that the interactive "foreman" and "worker" nodes have already been setup and are running, and the pipelines have been cloned from GitHub. If this is not the case, start here.

4.2.1 Retrieve data

Navigate to your RGB directory, download the data from the CyVerse DataStore with iRODS commands and untar:

```
cd /<personal_folder>/PhytoOracle/StereoTopRGB
iget -rKVP /iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/
↪stereoTop/<stereoTop-date.tar>
tar -xvf <stereoTop-date.tar>
```

Note: For a full list of available unprocessed data navigate to https://datacommons.cyverse.org/browse/iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/StereoTopRGB/

4.2.2 Retrieve vector and ML model files

Dowload the coordiate correction `.csv` file:

```
iget -N 0 -PVT /iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/
↪season10_multi_latlon_geno.geojson

iget -N 0 -PVT /iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/
↪necessary_files/gcp_season_10.txt

iget -N 0 -PVT /iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/
↪necessary_files/model_weights.pth
```

4.2.3 Edit scripts

- `process_one_set.sh`, `process_one_set2.sh`

  Find your current working directory using the command `pwd`. Open `process_one_set.sh` and paste the output from `pwd` into line 14 (line 12 in `process_one_set2.sh`). It should look something like this:

  ```
  HPC_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/StereoTopRGB/"
  ```

  Set your `.simg` folder path in line 15 (line 13 in `process_one_set2.sh`).

  ```
  SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_images/"
  ```

- `run.sh`

  +Open `run.sh` and paste the output from `pwd` into line 7. It should look something like this:

  ```
  PIPE_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/StereoTopRGB/"
  ```

  +Set your `.simg` folder path in line 8.

```
SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_
↪images/"
```

- `entrypoint.sh`, `entrypoint-2.sh`

  In lines 7 and 11, specify the location of CCTools:

  ```
  /home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/jx2json
  ```

  and

  ```
  /home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/makeflow
  ```

4.2.4 Run pipeline

Begin processing using:

```
./run.sh <folder_to_process>
```

Note: This may return a notice with a "FATAL" error. This happens as the pipeline waits for a connection to DockerHub, which takes some time. Usually, the system will fail quickly if there is an issue.

If the pipeline fails, check to make sure you have a "/" concluding line 14 of `process_one_set.sh`. This is one of the most common errors and is necessary to connect the program scripts to the HPC.

4.2.5 Troubleshooting and Issues

If problems arise with this pipeline, please refer to the tutorial on GitHub specific to the RGB pileline. If problems persist, raise an issue.

CHAPTER 5

Running the FlirIr Pipeline for Infrared Data

This pipeline extracts temperature data from image files. This guide provides demo data you can use follow along with and ensure the pipeline is functional. Before starting, change to `master` branch with `git checkout master`.

5.1 Pipeline Overview

FlirIr currently uses 7 different programs for data conversion:

Program	Function	Input	Output
flir2tif	Temperature calibrated transformer that converts bin compressed files to tif	`image.bin`, `metadata.json`	`image.tif`
collect_gps	Collects GPS coordinates from all geotiff files	`image.tif`	`collected_coordinates.csv`
Orthomosaicing	Finds best possible coordinates of all geotiffs	`collected_coordinates.csv`	`corrected_coordinates.csv`
replace_gps	Applies corrected GPS coordinates to images	`corrected_coordinates.csv`, `image.tif`	`corrected_image.tif`
flir_field_stitch	GDAL based transformer that combines all immages into a single orthomosaic	Directory of all converted `image.tif`	`ortho.tif`
plotclip	Clips plots from orthomosaic	`coordinatefile.geojson`, `ortho.tif`	`clipped_plots.tif`
flir_meantemp	Extracts temperature using from detected biomass	`coordinatefile.geojson`, Directory of all `clipped_plots.tif`	`meantemp.csv`

5.2 Running the Pipeline

Note: At this point, we assume that the interactive "foreman" and "worker" nodes have already been setup and are running, and the pipelines have been cloned from GitHub. If this is not the case, start here.

5.2.1 Retrieve data

Navigate to your directory containing FlirIr, and download the data from the CyVerse DataStore with iRODS commands and untar:

```
cd /<personal_folder>/PhytoOracle/FlirIr
iget -rKVP /iplant/home/shared/terraref/ua-mac/raw_tars/demo_data/Lettuce/FlirIr_demo.
↪tar
tar -xvf FlirIr_demo.tar
```

Data from the Gantry can be found within `/iplant/home/shared/terraref/ua-mac/raw_tars/season_10_yr_2020/flirIrCamera/<scan_date>.tar`

5.2.2 Edit scripts

- `process_one_set.sh`

  Find your current working directory using the command `pwd` Open `process_one_set.sh` and copy the output from `pwd` into line 14. It should look something like this:

  ```
  HPC_PATH="xdisk/group_folder/personal_folder/PhytoOracle/FlirIr/"
  ```

  Set your `.simg` folder path in line 8.

  ```
  SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_images/"
  ```

- `run.sh`
  - Paste the output from `pwd` into line 7. It should look something like this:

    ```
    PIPE_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/FlirIr/"
    ```

  - Set your `.simg` folder path in line 8.

    ```
    SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_images/
    ↪"
    ```

  - In line 4, specify the `<scan_date>` folder you want to process. For our purposes, this will look like:

    ```
    DATE="FlirIr_demo"
    ```

  - In lines 25 and 28, specify the location of CCTools:

    ```
    /home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/jx2json
    ```

    and

    ```
    /home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/makeflow
    ```

5.2.3 Run pipeline

Begin processing using:

```
./run.sh
```

Note: This may return a notice with a "FATAL" error. This happens as the pipeline waits for a connection to DockerHub, which takes some time. Usually, the system will fail quickly if there is an issue.

If the pipeline fails, check to make sure you have a "/" concluding line 14 of `process_one_set.sh`. This is one of the most common errors and is necessary to connect the program scripts to the HPC.

CHAPTER 6

Running the PSII Pipeline for Photosynthetic Potential Data

This pipeline uses the data transformers to extract chlorophyll fluorescence data from image files. Before starting, change to `alpha` branch with `git checkout alpha`.

6.1 Pipeline Overview

PSII currently uses 6 different programs for the analytical pipeline:

Program	Function	Input	Output
cleanmetadata	Cleans gantry generated metadata	`metadata.json`	`metadata_cleaned.json`
bin2tif	Converts bin compressed files to geotiff	`image.bin`	`image.tif`
resizetif	Resized original geotiffs to correct	`image.tif`	`resized_image.tif`
plotclip	Clips geotiffs to the plot	`resized_image.tif`, `shapefile.geojson`	`plot.tif`
psii_segmentation	Segments images given a validated set of thresholds	`plot.tif`	`segment.csv`
psii_fluorescence_aggregates	Aggregates segmentation data for each image and calculates F0, Fm, Fv, and Fv/Fm	`segment.csv`, `multitresh.json`	`:fluorescence_agg.csv`

6.2 Running the Pipeline

Note: At this point, we assume that the interactive "foreman" and "worker" nodes have already been setup and are running, and the pipelines have been cloned from GitHub. If this is not the case, start here.

6.2.1 Retrieve data

Navigate to your PSII directory, download the data from the CyVerse DataStore with iRODS commands and untar:

```
cd /<personal_folder>/PhytoOracle/FlirIr
iget -rKVP /iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/ps2Top/
↪<ps2Top-date.tar>
tar -xvf <ps2Top-date.tar>
```

Note: For a full list of available unprocessed data navigate to https://datacommons.cyverse.org/browse/iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/ps2Top/

6.2.2 Edit scripts

- `process_one_set.sh`, `process_one_set2.sh`

  Find your current working directory using the command `pwd`. Open `process_one_set.sh` and paste the output from `pwd` into line 15. It should look something like this:

  ```
  HPC_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/PSII/"
  ```

  Set your `.simg` folder path in line 16.

  ```
  SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_images/"
  ```

- `entrypoint.sh`, `entrypoint-2.sh`

  In lines 7 and 11, specify the location of CCTools:

  ```
  /home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/jx2json
  ```

  and

  ```
  /home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/makeflow
  ```

6.2.3 Run pipeline

Begin processing using:

```
./run.sh <folder_to_process>
```

Note: This may return a notice with a "FATAL" error. This happens as the pipeline waits for a connection to DockerHub, which takes some time. Usually, the system will fail quickly if there is an issue.

If the pipeline fails, check to make sure you have a "/" concluding line 14 of `process_one_set.sh`. This is one of the most common errors and is necessary to connect the program scripts to the HPC.

CHAPTER 7

Running the Scanner3DTop Pipeline for Generating 3D Point Clouds

This pipeline combines left and right 3D point clouds to create a single, merged 3D point cloud per range. Before starting, change to `master` branch with `git checkout master`.

7.1 Pipeline Overview

Scanner3DTop currently uses only a single distributed program:

Program	Function	Input	Output
3D MergePly	Merges PLY files into a single 3D point cloud	`left.ply`, `right.ply`	`merged.ply`

7.2 Running the Pipeline

Note: At this point, we assume that the interactive "foreman" and "worker" nodes have already been setup and are running, and the pipelines have been cloned from GitHub. If this is not the case, start here.

7.2.1 Retrieve data

Navigate to your directory containing Scanner3DTop, and download the data from the CyVerse DataStore with iRODS commands and untar:

```
cd /<personal_folder>/PhytoOracle/FlirIr
iget -rKVP /iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/
↪Scanner3DTop/<Scanner3DTop-date.tar>
tar -xvf <Scanner3DTop-date.tar>
```

Note: For a full list of available unprocessed data navigate to https://datacommons.cyverse.org/browse/iplant/home/shared/phytooracle/season_10_lettuce_yr_2020/level_0/Scanner3DTop/

7.2.2 Edit scripts

- `process_one_set.sh`

  Find your current working directory using the command `pwd` Open `process_one_set.sh` and copy the output from `pwd` into line 12. It should look something like this:

```
HPC_PATH="xdisk/group_folder/personal_folder/PhytoOracle/Scanner3DTop/"
```

  Set your `.simg` folder path in line 13.

```
SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_images/"
```

- `run.sh.main`
  - Paste the output from `pwd` into line 7. It should look something like this:

```
PIPE_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/Scanner3DTop/"
```

  - Set your `.simg` folder path in line 8.

```
SIMG_PATH="/xdisk/group_folder/personal_folder/PhytoOracle/singularity_images/
↪"
```

  - In line 4, specify the `<scan_date>` folder you want to process. For our purposes, this will look like:

```
DATE="<scan_date>"
```

  - In lines 16 and 19, specify the location of CCTools:

```
/home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/jx2json
```

    and

```
/home/<u_num>/<username>/cctools-<version>-x86_64-centos7/bin/makeflow
```

7.2.3 Run pipeline

Begin processing using:

```
./run.sh.main
```

Note: This may return a notice with a "FATAL" error. This happens as the pipeline waits for a connection to DockerHub, which takes some time. Usually, the system will fail quickly if there is an issue.

If the pipeline fails, check to make sure you have a "/" concluding line 14 of `process_one_set.sh`. This is one of the most common errors and is necessary to connect the program scripts to the HPC.