SAM Segmentation
Docker Guide

February 2025

1 Summary

This is a guide to a custom Segment Anything Model (SAM) program used to segment aerial RGB orthomosaics of hemp. The multiple dependencies and scripts involved have been compiled into a Docker image to simplify usage and promote batch processing. The Docker image should be able to run on a wide range of NVIDIA GPUs. A bash script (.sh file) is also provided to run these scripts from outside the Docker image and produce results.

The Docker image consists of a series of Python scripts which are called upon within a running instance of the Docker image. The scripts called directly are:

- **roi_selection.py**: Generates an RGB view of the orthomosaic and allows the user to interactively crop the image to a smaller region of interest
- **dbscan_seg_plant.py**: Detects individual plants based on the DBScan algorithm and generates bounding boxes for each plant.
- **extract_plants.py**: Segments the original orthomosaic to isolate crop pixels, generating RGB images of the individual plants (no background pixels). Can be executed with or without SAM.

2 Setup

For Windows users:

- Install and Set-up WSL2 (Windows Subsystem for Linux)
- Install a Linux distro (preferably Ubuntu)
- Make sure a compatible NVIDIA driver is installed
- Install versions of CUDA Toolkit and CUDNN compatible with your NVIDIA GPU:

  - First, check your GPU name and go to `https://developer.nvidia.com/cuda-gpus#compute` to find its compute capability
  - Navigate to `https://en.wikipedia.org/wiki/CUDA#GPUs_supported` to check which CUDA toolkit is compatible with your GPU.
  - Find toolkit on NVIDIA website and install.
  - Check `https://docs.nvidia.com/deeplearning/cudnn/backend/latest/reference/support-matrix.html` and select CUDNN version to install based on the version of the CUDA toolkit installed.

- Install Docker Desktop
- Ensure Docker Desktop uses WSL2 as backend:
  - After installation, open Docker Desktop
  - Navigate to "Settings"
  - From the General tab, check if the "Use WSL 2 based engine" option is selected. If not, select it and then select "Apply and restart"
- Download custom SAM Docker image and the bash script that accompanies it.

  Docker image: `segment-anyhemp_4.0_cuda12.1.tar`

  bash script: `Segment-anything_updated_cuda12.1.sh`
- Make sure Docker engine is running
- Use the Docker `load` command to load the image into Docker:

  `docker load --input filename.tar`

  via powershell, Ubuntu terminal or cmd
- Ensure the package x11-apps is installed in WSL. This can be done by executing:

  `sudo apt update`

  and `sudo apt install x11-apps`

  in the WSL terminal

For Linux users:

- Make sure a compatible NVIDIA driver is installed
- Install compatible versions of the CUDA Toolkit and CUDNN as well as any dependencies needed
- Install Docker Desktop for Linux

- Download custom SAM Docker image and the bash script that accompanies it.

  Docker image: `segment-anyhemp_4.0_cuda12.1.tar`

  bash script: `Segment-anything_updated_cuda12.1.sh`

- Make sure Docker engine is running

- Use the Docker `load` command to load the image into Docker:

  ```
  docker load --input segment-anyhemp_4.0_cuda12.1.tar
  ```

  via terminal.

- For some Linux users, Docker does not have access to the directory required to use the display. This can be accomplished by adding `/tmp/.X11-unix` to Docker resources. Open Docker, go to **Resources**, click on the **File Sharing** option and add: `/tmp/.X11-unix`.

3 Using the Docker image and bash script

To use the Docker image via the bash script for Linux or Windows:

- Start up the Docker Engine by opening Docker Desktop

- Ensure the engine is running

- Open the Linux Terminal (for Windows open the terminal of the WSL distro installed, e.g. Ubuntu)

- Navigate to the location of the bash script and run the bash script. The bash script is run by typing:

  ```
  bash Segment-anything_updated_cuda12.1.sh --arg1 --arg2....
  ```

Details on how to configure the script are given in the next section.

3.1 Running the bash script

As shown above, the bash script can be configured with certain arguments, some of which are required, depending on what script(s) within the Docker image you want to run.

Note that there is a primary image with which the other dates are coregistered. For purposes of the file structure below, the primary image's file name is `yp_mp_dp_RGB.tif` where `yp` = year, `mp` = month, and `dp` = date

Before executing the bash script, ensure that your image files are organized according to the following structure:

```
Your_project
└─ RGBmaps (contains GEOTIFF orthomosaics)
```

```
├── yp_mp_dp_RGB.tif.  (this is the primary image)
├── yy_mm_dd_RGB.tif
├── yy_mm_dd_RGB.tif
├── ...
└── Coregistered (contains co-registered GEOTIFF orthomosaics)
    ├── yy_mm_dd_to_yp_mp_dp_RGB.tif
    ├── yy_mm_dd_to_yp_mp_dp_RGB.tif
    └── yy_mm_dd_to_yp_mp_dp_RGB.tif
```

- The parent folder (`Your_project`) can have any name.
- The sub-folder for the RGB orthomosaics **must** be named: `RGBmaps`
- Likewise, if you choose to co-register your images, the sub-subfolder for the co-registered images **must** be named: `Your_project\RGBmaps\Coregistered`
- The orthomosaic files **must** be named in this format: `yy_mm_dd_RGB.tif` where `yy` = year, `mm` = month, and `dd` = date

Note: Not all arguments are required for each function. Some functions require fewer arguments and some arguments have default values and may not need further input. The arguments are listed below:

- **-indir/--input-directory**: Path to the parent folder.

  Ex: `\Desktop\Your_project`
- **-map/–map-name**: The first part of the chosen image's filename, which is also a date. Ex: `22_08_10`
- **-dates/--other-dates**: Dates for additional image files in the `RGB_maps` folder. Ex: `22_08_18 22_08_20`
- **-prog/--prog-value**: This argument must be a number. This number determines which script(s) to execute. See the Functions section for prog-value specifications.
- **-sam/–sam-segment**: Turns on sam-segmentation in the plant extraction function. Default is `off`
- **-w/--width**: Sets width (in pixels) of boundary box in DBScan. Default is `160`
- **-h/--height**: Sets height (in pixels) of boundary box in DBScan. Default is `160`
- **-exg_th/--exg_threshold**: Sets the Excess Green Index (ExG) threshold in DBScan. Must be a number between 0 - 1.0. Default is `0.2`
- **-kernel/--**`kernel_size`: Sets the kernel size in DBScan. Default is `5`

- **-min/–min-counts**: Parameter for the exg calculation. Default is `0`
- **-field_col/--field_num_col**: Number of plant columns in the field. This can be used to help assign the correct IDs to each plant. In gridded planting systems, this is equivalent to the number of plants per row. Default value is `50`
- **-f_id/--first_id**: The first plant ID in the list of all plant IDs. This depends on the user's plant identification system and assumes that plants are numbered. However, if plants do not have numerical identifiers the user can either use the default value (numbering starting at 3000) or choose a different number to start with (ex: 0 or 1). Default is `3000`
- **--row_col/--field_num_row**: Number of plant rows in the field. Also helps in assigning correct id's to each plant. Same as number of plants per column. Default is 15

4 Functions

Note: The functions have to be run in the order in which they are listed as the successful completion of one provides the data for the function after it.

- **ROI selection**: Prompts the user to select a region of interest (ROI) in the .tif file for more accurate identification and segmentation of individual plants. Its **-prog** value is **1**. When run, a window appears containing the RGB orthomosaic. Use the left mouse button to place points on the image to set the new boundary. Press the middle mouse button and then the escape button (**in that order**) when you have finished placing points. A right mouse click is used to undo a point in the boundary. Make sure to select a tight boundary (leave out any unnecessary portions of the field displayed in the image).

  Some errors may appear in the terminal during execution. These errors can be ignored if the image loads. Due to the dimensions of the image, the window tends not to fit on screen and requires some resizing to fit the whole orthomosaic.

  Required arguments: `-indir -map`

- **DBScan**: Used to automatically identify different plants based on an ExG threshold value and generate bounding boxes for each plant. Its **-prog** value is **2**.

  Required arguments: `-indir -map -field_col -f_id -row_col`

  Optional arguments: `-w -h -exg_th -kernel -min`

- **Co-registration**: After executing ROI selection and DBScan for one image, the user can select other images acquired over the same area and co-register them with the first image. This enables the bounding boxes generated for the first image to be re-used for the other images, eliminating the need to perform the ROI selection and DBScan steps on each individual image. The co-registration **-prog** value is **4**.

Required arguments: `-indir -map -dates`

*`-map` = primary orthomosaic. This is the orthomosaic to which the other dates are coregistered.

* In your `-dates` argument, leave out the primary orthomosaic date. You can co-register multiple dates at once, provided the arguments you pass to `-dates` are in quotes. In the following example, `22_08_10` is the primary orthomosaic. Example usage:

```
bash Segment-anything_updated_cuda12.1.sh -indir Desktop/my_project
-map 22_08_10 -prog 4 -dates "22_08_03 22_08_18"
```

- **Plant Extraction**: Extracts the individual plants from the primary orthomosaic (orthomosaic to which the other dates are coregistered) as well as the other co-registered dates (if applicable). Its **-prog** value is **3**.

  Required arguments: `-indir -map -dates -sam`

  To extract segmentation results for the primary orthomosaic, it must be set as the `-map` argument **and** be included in the `-dates` argument. In the following example, `22_08_10` is the primary orthomosaic. Example usage:

```
bash Segment-anything_updated_cuda12.1.sh -indir /Desktop/my_project
-map 22_08_10 -prog 3 -dates "22_08_10 22_08_03 22_08_18" -sam on
```

  As noted above, `-dates` can take multiple arguments provided they are in quotes.

  If you only want to extract segmentation results for the primary orthomosaic, use the same argument for both `-map` and `-dates`. Example usage:

```
bash Segment-anything_updated_cuda12.1.sh -indir /Desktop/my_project
-map 22_08_10 -prog 3 -dates "22_08_10" -sam on
```

  In all cases, the `-sam` flag argument should be on, ie. `-sam on`

5 Logging

To log each process for later reference, one can store the terminal outputs in a .txt file. This can be done by typing:

```
bash Segment-anything_updated_cuda12.1.sh -arg1 -arg2 -arg3.. 2>&1
| tee log.txt
```