SAM Segmentation
Docker Guide

February 2025

1 Summary

This is a guide to a custom Segment Anything Model (SAM) program used to segment aerial orthomosaics of hemp and grapevine. 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

- Install Docker Desktop
- Ensure Docker Desktop uses WSL2 as backend
- Download custom SAM image and the bash script that accompanies it
- 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 image and the bash script that accompanies it
- Make sure Docker engine is running
- Use the Docker `load` command to load the image into Docker:

  `docker load --input filename.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.

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

```
Your_project
└── RGB_maps (contains GEOTIFF orthomosaics)
    ├── mm_dd_yy_RGB.tif
    ├── mm_dd_yy_RGB.tif
    ├── mm_dd_yy_RGB.tif
    ├── ...
    └── Coregistered (contains co-registered GEOTIFF orthomosaics)
        ├── mm_dd_yy_RGB_coreg.tif
        ├── mm_dd_yy_RGB_coreg.tif
        └── mm_dd_yy_RGB_coreg.tif
```

- The parent folder (`Your_project`) can have any name.
- The sub-folder for the RGB orthomosaics **must** be named: `RGB_maps`
- Likewise, if you choose to co-register your images, the sub-subfolder for the co-registered images **must** be named: `Your_project\RGB_maps\Coregistered`
- The orthomosaic files **must** be named in this format: `mm_dd_yy_RGB.tif`

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: `08_16_22`
- **-dates/--other-dates**: Dates for additional image files in the `RGB_maps` folder. Ex: `08_18_22 08_20_22`

- **-prog/--prog-value**: This argument must be a number. This number determines which script(s) to execute.
- **-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**
- **-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.
- **-f_id/--first_id**: The first plant ID in the list of all plant IDs. Ex: `Plant_3063`

4 Functions

- **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**.

  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`

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

- **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**.

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

Required arguments: `-indir -map`

Optional arguments: `-dates -sam`

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
```