OpenDroneMap

On this page, we are explaining and documenting the installation and usage of OpenDroneMap (ODM) on a Google Cloud Machine. The source code of ODM can be found in this GitHub Repo: GitHub OpenDroneMap

Installation

We are installing ODM on a Google Machine which is provided by the UNICAES. It is only turned on when needed.

As some prerequisites: We also configured a DNS entry to point to the static external IP of our Google Cloud Machine, so that we can get an SSL certificate for secure communications. The machine itself has 16 cores and 128GB of RAM with an additional 500GB of persistent storage.

Install Docker and Docker Compose

We followed the official documentation: Docker install - Using the repository

Getting WebODM

Download the git repository:

```
git clone https://github.com/OpenDroneMap/WebODM --config
core.autocrlf=input --depth 1
cd WebODM
```

Then WebODM can be started with the following:

```
./webodm.sh restart --ssl --hostname webodm.myorg.com
```

Preparing Drone Images from DJI Mavic 3 Multispectral

After taking the photos with the DJI M3M we need to do some postprocessing to be able to use them in ODM. Therefore it is necessary to split the multispectral from the plain RGB images. Also, the timestamps of the photos need to be matched just in case and the names need to be slightly modified as there were some hick ups in ODM from time to time. We already prepared a script for that:

correct_filenames.sh

```
#!/bin/bash

# Process images in the specified directory
```

```
process_images() {
    local target_dir="$1"

    mkdir "${target_dir}/RGB"
    mkdir "${target_dir}/MS"

    echo "Processing images in: $target_dir"

    for first_image in $(ls "$target_dir" | grep -E
'DJI_[0-9]{14}_[0-9]{4}_D\.JPG'); do
        echo "Found first image: $first_image"

        timestamp=$(echo "$first_image" | sed -E
's/DJI_([0-9]{14})_[0-9]{4}_D\.JPG/\1/')
        echo "Extracted timestamp: $timestamp"

        # Extract the number (XXXX) from the first image filename
        number=$(echo "$first_image" | sed -E
's/DJI_[0-9]{14}_([0-9]{4})_D\.JPG/\1/')
        echo "Extracted number: $number"

    mv "$target_dir/$first_image" "$target_dir/RGB/DJI-${timestamp}-
${number}_D.JPG"

        for ext in "MS_G" "MS_NIR" "MS_R" "MS_RE"; do
            echo "Searching for matching ${ext}.TIF files with number:
$number"

            image_to_rename=$(ls "$target_dir" | grep -E
"DJI_[0-9]{14}_${number}_${ext}\.TIF" | head -n 1)

            if [ -n "$image_to_rename" ]; then
        new_ext=$(echo "${ext}" | sed 's/_//')
                new_name="DJI-${timestamp}-${number}_${new_ext}.TIF"
                echo "Renaming $target_dir/$image_to_rename to
$target_dir/$new_name"
                mv "$target_dir/$image_to_rename"
"$target_dir/MS/$new_name"
            else
                echo "No matching ${ext}.TIF files found for number:
$number"
            fi
        done
    done
}

# Check if an argument is provided
if [ $# -ne 1 ]; then
    echo "Usage: $0 <folder_path>"
    exit 1
```

```
fi

# Call the function to process images
process_images "$1"
```

Just save this file somewhere you will be able to find it later as well.

We are running this script on Ubuntu. Do make it executable Don´t forget to:

```
sudo chmod +x ./correct_filenames.sh
```

You can then execute the script by giving the folder with pictures inside as an argument. Example:

```
./correct_filenames.sh ./pictures
```