Forest Green-up Monitoring — Dataset Imagery Quality Control Procedure v1.0

Last updated: 2025-10-09 - Alexander Bleasdale

1) Project Overview

Aim: Train a semantic segmentation model to classify **tree species** and **ground cover vegetation** in camera trap imagery for forest green-up monitoring.

Target output: Semantic segmentation of tree species to derive ROIs on individual species canopies. The model is **not** required to identify individual trees.

Data characteristics: Multi-location time series; varying canopy density; heavy snow cover, near-infrared imagery, occluded imagery, low light levels.

2) Dataset Acquisition

Camera Trap Hardware:

Nordmaling Dataset “Basic-classificationNM” – Reconyx

3) Dataset Selection and Download

The following datasets were downloaded from **Trapper** through **Jupyter**

Nordmaling

Select images that have been classified as blank- “May be redundant”

Select images that are taken from a timelapse acquisition strategy. As there is no category that denotes timelapse or motion-controlled acquisition this can be selected through acquisition time.

In Nordmaling timelapse imagery is captured at **12:00:00 pm**

4) Image Naming Protocol

Images currently have generic names *‘IMG_XXX’* which can repeat between camera trap deployment sites. Renaming images to include location, deployment site, and date creates unique and informative names that allows for efficient data management.

LC_DS_20XX-XX-XX

Where LC is Location Code, DS is Deployment Site, and Date is in Year-Month-Day format

5) Header and Footer Removal

The inclusion of camera trap header and footers can influence the quality control procedures and impact classification, and removal is recommended. Removal of the header and footer is essential for running the image quality control scripts located below.

Camera Trap Hardware	Header Height	Footer Height
Reconyx	32 pixels	64 pixels

If the camera trap hardware is not listed here, measure the height of the header and footer in pixels using an image editing software.

The script for header and footer removal is located here ...

6) Image Quality Control

The image quality control protocol is implemented through a single modular Python script that can be adjusted to suit different datasets, environmental conditions, or research goals. The script processes images from a given folder (including subfolders), automatically identifying and removing those that fail to meet quality criteria. Rejected images are moved to a separate folder named Quality Control for later inspection.

The following quality control checks are performed within the script:

6.1) Near-Infrared Imagery

Under overcast conditions or during winter months in dense forest environments, camera traps may capture imagery in the near-infrared (NIR) spectrum, often presented in grayscale. These images are unsuitable for colour-based vegetation phenology analysis.

To detect and exclude such images, the script analyses a horizontal strip (typically 1000 pixels in height) across the central region of each image. Two conditions may trigger classification as NIR:

- The image is fully grayscale (i.e., lacks colour channels).
- The image exhibits **low saturation**, defined as a saturation value below 20 across most of the examined region.

These checks ensure that only true RGB images with sufficient colour information are retained for analysis.

6.2) Blurry and Dark Imagery

Blurred or dark canopy images offer limited value for phenological assessment due to a lack of structural or colour detail.

The script applies a **Laplacian variance filter** to quantify image sharpness. Images with low variance are flagged as blurred, which may result from:

- Lens condensation,
- Persistent fog or mist,
- Raindrops on the lens,
- Motion blur or defocus.

In addition, brightness and contrast thresholds are used to detect and remove images that are significantly underexposed (dark), typically defined by low mean intensity and low standard deviation in grayscale pixel values.

Both blur and darkness filters are customizable through threshold parameters, allowing users to tailor sensitivity based on expected conditions in the dataset

See section **def detect_blur**

6.3) Sensor Occlusion

In snowy environments, it is common for camera traps to become partially occluded by snow accumulating on the lens or protective housing. This can result in a distinct, **uniform grey or white band,** typically in the lower portion of the image, which obstructs visibility of the scene and renders the data unusable for vegetation analysis.

To automatically detect and remove such images, the script performs the following operations on the **bottom 25% of each image**:

- Calculates the **standard deviation** of grayscale pixel values to assess texture and variation. Occluded regions typically show **very low variation** due to their uniform colour.
- Computes the **row-wise standard deviation** to detect horizontal uniformity (i.e., a consistent band) and flags the image if a high proportion of rows are flat.
- Measures the **mean brightness** to distinguish snow occlusion from shadows; only bright, low-variation regions are considered snow blockage.
- Checks for the presence of **green pixels** in the bottom region to avoid false positives caused by flat but natural vegetation scenes.

Images that meet all criteria — low texture, high flatness, sufficient brightness, and low green content — are classified as occluded by snow and moved to the Quality Control folder.

This method is effective at identifying obstructions caused by snow build-up directly on the lens or housing, ensuring that only fully visible scenes are retained for analysis

Section **def detect_snow_blockage**

6.4) Low Saturation Imagery (Unless Vegetation is Present)

Low-saturation images typically from heavy overcast skies, fog or heavy snow are typically unsuitable for colour-based vegetation analysis. These images may result from poor lighting, sensor errors, or atmospheric interference. However, in some cases, valid scenes containing **naturally green vegetation** may also exhibit low overall saturation.

To avoid removing useful images, this stage performs a conditional saturation check:

- The script first evaluates the **proportion of low-saturation pixels** across the entire image (pixels where saturation is below a threshold, e.g., 30).
- If a large portion of the image is low in saturation (e.g., >90%), it is flagged for potential removal.
- Before removal, the script assesses the **top half of the image** for the presence of green pixels, using hue and saturation thresholds in HSV space. If the top region contains a significant proportion of green (e.g., >40%), the image is preserved.

This approach effectively removes washed-out or fog-obscured images, while retaining valid scenes with low overall colour contrast but visible vegetation.

Section **def detect_low_saturation**

6.5) Low Saturation Imagery (Unless Vegetation is Present

Heavy snow cover can also create scenes dominated by a blue-tint, meaning the previous low-saturation stage can be ineffective.

To detect such conditions:

- The bottom 25% of the image is analysed for **blue-dominant pixels**, defined as pixels where the blue channel significantly exceeds both red and green (e.g., B > R + 20 and B > G + 20).
- If the proportion of blue-dominant pixels exceeds a defined threshold (e.g., 50%), the image is flagged as snow-dominated.
- However, the script then checks the **top half of the image** for green canopy using HSV colour thresholds. If the top region contains sufficient green vegetation (e.g., >40%), the image is considered valid and retained.

This method ensures that snow-dominated images without visible vegetation are filtered out, while allowing mixed scenes with snow-covered ground but visible canopy to pass quality control.

Section **def detect_snow_by_blue_pixels**

---

7) Validation

8) Quick Reference

a) The quality control automatically filters and removes low-quality or unsuitable time-lapse camera images into a Quality_Control folder for review or exclusion
b) The procedure works on folders and subfolders and moves all images not passing quality checks to a new directory within the original structure
c) Rename all images prior to the quality control procedure
d) Remove the camera trap headers and footers (and logo) prior to the quality control procedure
e) **NIR Detection** detects grayscale or low-saturation images likely captured in near-infrared mode.
f) **Blur Detection** uses Laplacian variance to detect lack of sharpness or condensation blur.
g) **Snow Blockage** detection **i**dentifies uniform bright regions at the bottom of the image that suggest camera occlusion (e.g. snow pile-up).
h) **Low Saturation** detects desaturated images unless the low-saturation areas are predominantly green (e.g. healthy canopy).
i) **Heavy Snow Detection (Blue Pixels)** identifies snow by blue-dominant pixels in the bottom half, unless the top half is mostly green (to avoid false positives).
j) Perform the quality control before annotating imagery

Parameter	Section	Description	Suggested Value
saturation > 20 (central strip)	6.1 - Near-Infrared Detection	Min saturation in centre to consider RGB	20
laplacian_threshold	6.2 - Blur Detection	Laplacian variance threshold to detect blur	150
std_threshold	6.3 - Snow Blockage	Low grayscale std indicates occlusion	12
flat_row_std	6.3 - Snow Blockage	Max row std to count row as flat	12
flat_row_ratio	6.3 - Snow Blockage	Proportion of flat rows to trigger blockage	0.6
min_brightness	6.3 - Snow Blockage	Brightness threshold to avoid shadow false positives	50
green_ratio_threshold	6.3 - Snow Blockage	If bottom is mostly green, skip blockage removal	0.5
saturation_threshold	6.4 - Low Saturation	Max sat to count as low-saturation pixel	100
low_sat_ratio_threshold	6.4 - Low Saturation	Proportion of image that must be low-sat to flag	0.75
green_allowance	6.4 - Low Saturation	Allowed green pixels in low-sat region	0.5
green_hue_range	6.4 / 6.5	HSV hue range used for green detection	(35, 85)
blue_ratio_threshold	6.5 - Blue-Dominant Snow	Blue-dominant pixel ratio to trigger snow flag	0.4
green_top_threshold	6.5 - Blue-Dominant Snow	Proportion of green in top half to skip	0.5

Quality Control Script

a) Header and Footer Removal

```
b) import os
   import cv2
   import numpy as np
   import shutil


   def detect_blur(image, threshold=150):
   gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
   laplacian_var = cv2.Laplacian(gray, cv2.CV_64F).var()
   return laplacian_var < threshold


   def detect_snow_blockage(image, std_threshold=12, flat_row_std=12,
   flat_row_ratio=0.6, min_brightness=50,
   green_ratio_threshold=0.5):
   height = image.shape[0]
   bottom = image[int(height * 0.9):, :, :]
   gray = cv2.cvtColor(bottom, cv2.COLOR_BGR2GRAY)


   std_dev = np.std(gray)
   mean_brightness = np.mean(gray)
   row_std_devs = np.std(gray, axis=1)
   flat_rows_ratio = np.sum(row_std_devs < flat_row_std) /
   len(row_std_devs)


   b, g, r = cv2.split(bottom)
   green_mask = (g > r + 20) & (g > b + 20)
   green_ratio = np.sum(green_mask) / green_mask.size


   if green_ratio > green_ratio_threshold:
   return False


   return (
   std_dev < std_threshold and
   flat_rows_ratio > flat_row_ratio and
   mean_brightness > min_brightness
   )


   def detect_low_saturation(image, saturation_threshold=100,
   low_sat_ratio_threshold=0.75,
   green_hue_range=(35, 85),
   green_allowance=0.5):
   hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
   h, s, _ = cv2.split(hsv)


   low_sat_mask = s < saturation_threshold
   low_sat_ratio = np.sum(low_sat_mask) / s.size


   if low_sat_ratio < low_sat_ratio_threshold:
   return False
```

```
green_mask = (h >= green_hue_range[0]) & (h <= green_hue_range[1])
green_in_low_sat = green_mask & low_sat_mask


if np.sum(low_sat_mask) == 0:
return False


green_ratio_in_low_sat = np.sum(green_in_low_sat) /
np.sum(low_sat_mask)
return green_ratio_in_low_sat < green_allowance


def detect_snow_by_blue_pixels(image, blue_ratio_threshold=0.4,
green_hue_range=(35, 85),
green_top_threshold=0.5,
verbose=False):
height = image.shape[0]
bottom_25 = image[int(height * 0.5):, :, :]
b, g, r = cv2.split(bottom_25)


blue_mask = (b > r + 20) & (b > g + 20)
blue_ratio = np.sum(blue_mask) / blue_mask.size if blue_mask.size > 0
else 0


if verbose:
print(f"Blue ratio (bottom 25%): {blue_ratio:.3f}")


if blue_ratio <= blue_ratio_threshold:
return False


hsv = cv2.cvtColor(image, cv2.COLOR_BGR2HSV)
top_half = hsv[:height // 2, :, :]
h_top, s_top, _ = cv2.split(top_half)


green_mask = (h_top >= green_hue_range[0]) & (h_top <=
green_hue_range[1]) & (s_top > 30)
green_ratio_top = np.sum(green_mask) / green_mask.size if
green_mask.size > 0 else 0


if verbose:
print(f"Green ratio (top half): {green_ratio_top:.3f}")


return green_ratio_top <= green_top_threshold
root_directory(r"path/to/your/dataset")
```