Image Quality Control Algorithms

Various approaches has been developed and revised in existing algorithms to build Image Quality Control module for *PhenoAI*. This supplementary report will pose as a detailed guide for users who are seeking the details of all the algorithms used for the classification and segregation of low-quality or unclear images.

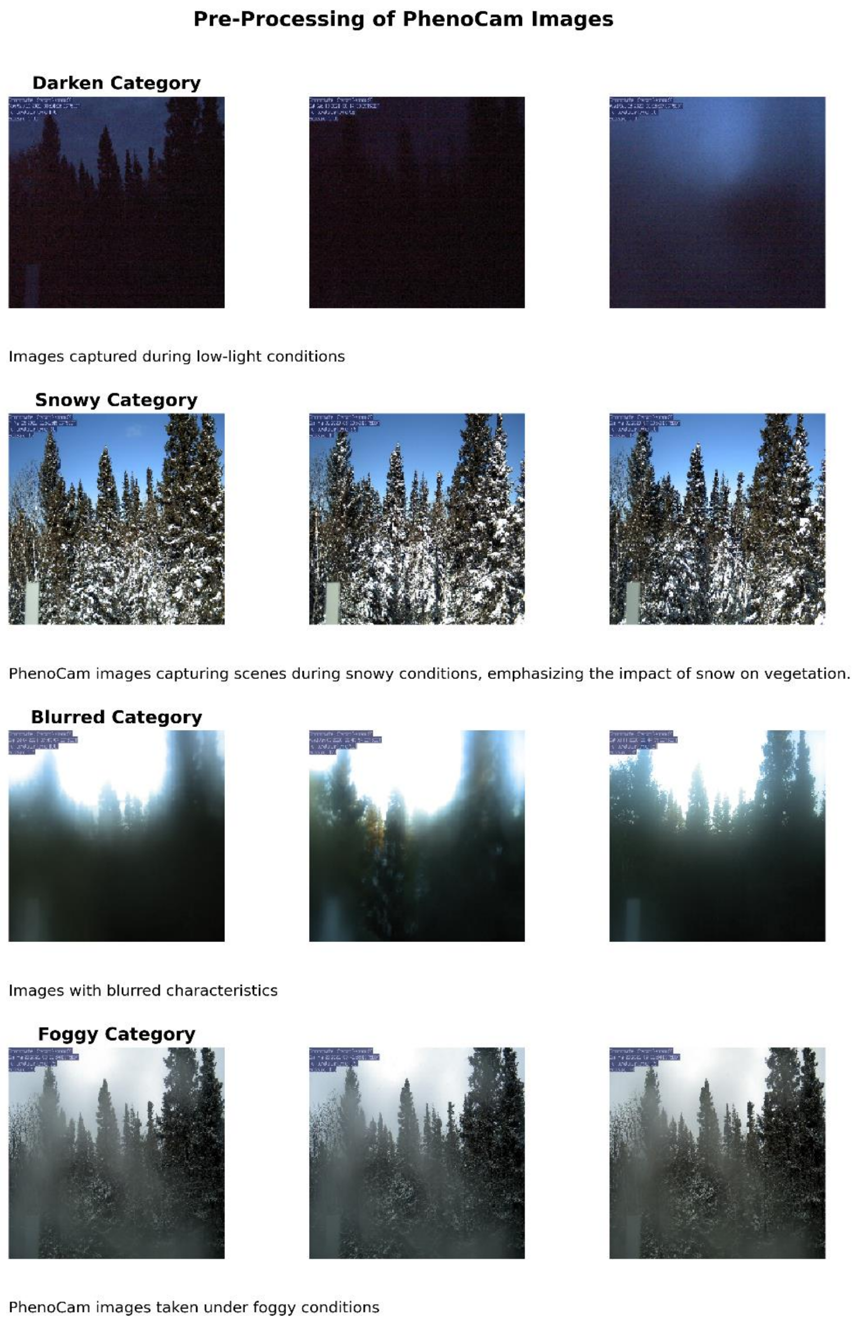
Pre-Processing of PhenoCam Images
Darken Category
Images captured during low-light conditions
Snowy Category
PhenoCam images capturing scenes during snowy conditions, emphasizing the impact of snow on vegetation.
Blurred Category
Images with blurred characteristics
Foggy Category
PhenoCam images taken under foggy conditions

Figure 1. Example of output

Here is the separate python code, that can be downloaded using github link. Users can use this code for testing on their phenocam imageries. Input file and output file path needs to be updated by the user.

https://github.com/ItsAkashPandey/PhenoAI/blob/main/Image_Quality_Control_final.ipynb

Explanation of Algorithms with mathematical background

1. Function for categorizing images as Foggy/Hazy

The function *detect_fog* uses several steps to identify foggy or hazy images. Here are the mathematical formulas and explanations for each step:

1.1 Convert the image to grayscale

This step converts a color image into a grayscale image by eliminating the hue and saturation information. Formula for this conversion is:

$$Y = 0.299R + 0.587G + 0.114B$$

Where Y is the grayscale intensity, and R,G, and B are the red, green, and blue values of the color image. This formula is based on the luminance coefficients of the standard RGB color space.

1.2 Apply *GaussianBlur* to reduce noise and enhance fog features

This step applies a Gaussian filter to the grayscale image to smooth out the noise and emphasize the fog features. A Gaussian filter is a low-pass filter that attenuates the high-frequency components of the image. A Gaussian filter can be represented by a kernel, which is a matrix of weights that are applied to each pixel and its neighbors. Formula for a Gaussian kernel is:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

Where *G(x,y)* is the weight at the position *(x,y)*, and σ is the standard deviation of the Gaussian distribution. The size of the kernel is determined by the filter radius, which is usually chosen as 3σ. The larger the filter radius, the more blurred the image becomes. This formula is based on the Gaussian function, which is a widely used probability distribution.

1.3 Compute the absolute difference between the original and blurred images

This step computes the absolute difference between the grayscale image and the blurred image. This difference highlights the edges of the image, which are usually less affected by fog. A formula for this difference is:

$$D(x,y) = |Y(x,y) - G(x,y)|$$

Where *D(x,y)* is the difference at the position *(x,y)*, and *Y(x,y)* and *G(x,y)* are the grayscale and blurred intensities at the same position. This formula is based on the absolute value function, which returns the magnitude of a number .

1.4 Threshold the difference image to identify foggy regions

This step applies a threshold to the difference image to create a binary mask that indicates the foggy regions. A threshold is a value that separates the pixels into two groups: those that are above the threshold and those that are below the threshold. A formula for this thresholding is:

$$M(x,y) = \begin{cases} 255, & if\ D(x,y) \leq T \\ 0, & otherwise \end{cases}$$

Where *M(x,y)* is the mask at the position *(x,y)*, and *T* is the threshold value. The mask has a value of 255 (white) for the foggy regions and 0 (black) for the non-foggy regions. The choice of the threshold value depends on the image characteristics and the desired sensitivity of the fog detection.

1.5 Additional features for fog detection

This step uses some additional features to refine the fog detection based on the image statistics. These features are:

1.5.1 Mean intensity:

This is the average pixel intensity of the grayscale image. It measures the overall brightness of the image. A formula for this feature is:

$$\mu = \frac{1}{N}\sum_{x,y} Y(x,y)$$

Where μ is the mean intensity, and N is the total number of pixels in the image. This formula is based on the arithmetic mean, which is a measure of central tendency.

1.5.2 Standard deviation

This is the standard deviation of the pixel intensities of the grayscale image. It measures the variation or contrast of the image. A formula for this feature is:

$$\sigma = \sqrt{\frac{1}{N}\sum_{x,y} (Y(x,y) - \mu)^2}$$

Where σ is the standard deviation, and μ is the mean intensity. This formula is based on the standard deviation, which is a measure of dispersion .

1.5.3 Mean blur

This is the average pixel intensity of the blurred image. It measures the degree of blurriness of the image. A formula for this feature is:

$$\beta = \frac{1}{N}\sum_{x,y} G(x,y)$$

Where β is the mean blur, and $G(x,y)$ is the blurred intensity. This formula is based on the arithmetic mean.

1.5.4 Adjust the fog detection based on mean intensity, standard deviation, and mean blur

This step adjusts the fog detection based on some empirical rules that are derived from the observation of foggy images. These rules are:

(1) The mean intensity of a foggy image is usually high, because fog reflects and scatters light. A possible rule is:

$$\mu > 100$$

where 100 is an arbitrary threshold that can be adjusted as needed.

(2) The standard deviation of a foggy image is usually low, because fog reduces the contrast and details of the image. A possible rule is:

$$\sigma < 20$$

where 20 is an arbitrary threshold that can be adjusted as needed.

(3) The mean blur of a foggy image is usually low, because fog makes the image look smooth and uniform. A possible rule is:

$$\beta < 90$$

User can adjust the threshold.

(4) The percentage of the fog mask that is white is usually high, because fog covers a large area of the image. A possible rule is:

$$\frac{\sum_{x,y} M(x,y)}{255N} > 0.1$$

User can adjust the threshold.

(5) The final fog detection is the logical *AND* (∧) of all these rules, which means that all of them must be satisfied for the image to be classified as foggy.

$$[\text{is\_foggy} = (\mu > 100)\sim \wedge \sim(\sigma < 20)\sim \wedge \sim(\beta < 90)\sim \wedge \sim\left(\frac{\sum_{x,y} M(x,y)}{255N} > 0.1\right)]$$

2. Function for categorizing images as Snowy

function *detect_snow*

2.1 Define a range for snow color in HSV

This step defines a range for the snow color in the HSV color space. Snow is usually white or light gray, which means that it has a high value and a low saturation. The hue component is not very important for snow, as it can vary depending on the lighting conditions. A possible range for snow color in HSV is:

$$0 \le H \le 180$$
$$0 \le S \le 0.1$$
$$0.7 \le V \le 1$$

Where *H*, *S*, and *V* are the hue, saturation, and value components of the HSV image. These values are arbitrary and can be adjusted as needed. This range is based on the observation of snow color in different images.

2.2 Create a mask to extract the lower vegetation part

This step creates a mask to extract the lower vegetation part of the image. The lower vegetation part is the region below the horizon, where the trees and plants are located. Snow is more likely to accumulate on the lower vegetation than on the sky. A possible way to create this mask is:

$$L(x,y) = \begin{cases} 0, & if\ y < \frac{H}{2} \\ 255, & otherwise \end{cases}$$

Where *L(x,y)* is the mask at the position *(x,y)*, and *H* is the height of the image. This mask has a value of 0 (black) for the upper half of the image and 255 (white) for the lower half of the image. This mask is based on the assumption that the in phenocam image, sky is in upper part of the image for roughly 20-40% of the area, which may not be true for all images.

2.3 Apply the mask to the HSV image

This step applies the mask to the HSV image to obtain the HSV image of the lower vegetation part. This can be done by using the bitwise AND operation, which returns the value of the first operand if the corresponding bit of the second operand is 1, and 0 otherwise. A formula for this operation is:

$$V'(x,y) = V(x,y) \wedge L(x,y)$$

Where *V'(x,y)* is the HSV image of the lower vegetation part at the position *(x,y)*, and *V(x,y)* and *L(x,y)* are the HSV image and the mask at the same position.

2.4 Create a mask for snow in the lower vegetation part

This step creates a mask for snow in the lower vegetation part by using the range for snow color in HSV. This can be done by using the *inRange* function, which returns 255 (white) if the input value is within the specified range, and 0 (black) otherwise. A formula for this function is:

$$S'(x,y) = \text{inRange}(V'(x,y), L_S, U_S)$$

Where *S'(x,y)* is the mask for snow in the lower vegetation part at the position *(x,y)*, and *V'(x,y)* is the HSV image of the lower vegetation part at the same position.L_S and U_S are the lower and upper bounds of the range for snow color in HSV, respectively.

2.5 Use morphology to enhance snow regions

This step uses morphology to enhance the snow regions in the mask. Morphology is a set of operations that modify the shape and structure of an image. One of the common morphology operations is closing, which is a combination of dilation and erosion. Dilation is an operation that expands the white regions in an image by adding pixels to the boundaries. Erosion is an operation that shrinks the white regions in an image by removing pixels from the boundaries. Closing is an operation that first dilates and then erodes an image, which can fill small holes and gaps in the white regions. A formula for closing is:

$$S''(x,y) = \text{erode}(\text{dilate}(S'(x,y), K), K)$$

Where *S''(x,y)* is the enhanced mask for snow in the lower vegetation part, and *S'(x,y)* is the original mask for snow in the lower vegetation part at the same position. *K* is a kernel, which is a matrix of weights that are used for the dilation and erosion operations.

2.6 Calculate the percentage of the lower vegetation covered by snow

This step calculates the percentage of the lower vegetation covered by snow by using the enhanced mask for snow. This can be done by dividing the number of white pixels in the mask by the total number of pixels in the mask. A formula for this calculation is:

$$P = \frac{\sum_{x,y} S''(x,y)}{255N}$$

Where P is the percentage of the lower vegetation covered by snow, and $S''(x,y)$ is the enhanced mask for snow in the lower vegetation part. N is the total number of pixels in the mask, which is equal to the area of the lower vegetation part.

The final snow detection is based on a threshold for the percentage of the lower vegetation covered by snow, which means that the image is classified as snowy if the percentage is above the threshold.

$$[\text{is\_snowy} = P > Threshold]$$

3. Function for categorizing images as Blurred

function *detect_blurred_phenocam*

This function uses the Laplacian operator to measure the amount of blur in an image. Here are some possible steps and formulas for this function:

3.1 Convert the image to grayscale

This step converts a color image into a grayscale image as described in the section 1.1.

3.2 Calculate the Laplacian of the image to detect edges

This step calculates the Laplacian of the image to detect the edges and contours of the image. The Laplacian is a second-order derivative operator that measures the change of gradient in an image. A formula for the Laplacian is:

$$L'(x, y) = \frac{\partial^2 Y}{\partial x^2} + \frac{\partial^2 Y}{\partial y^2}$$

Where *L'(x,y)* is the Laplacian at the position *(x,y)*, and *Y* is the grayscale intensity.

3.3 Calculate the variance of the Laplacian:

This step calculates the variance of the Laplacian, which is a measure of how spread out the Laplacian values are. The variance of the Laplacian reflects the amount of blur in the image, as a blurred image has less edges and contours, and thus less variation in the Laplacian. A formula for the variance of the Laplacian is:

$$V = \frac{1}{N} \sum_{x,y} (L'(x, y) - \mu)^2$$

Where *V* is the variance of the Laplacian, and *L'(x,y)* is the Laplacian, *N* is the total number of pixels in the image, and μ is the mean of the Laplacian, which is given by:

$$\mu = \frac{1}{N} \sum_{x,y} L(x, y)$$

3.4 Apply a threshold to detect blurred images

This step applies a threshold to the variance of the Laplacian to classify the image as blurred or not. A threshold is a value that separates the images into two groups: those that have a variance of the Laplacian below the threshold and those that have a variance of the Laplacian above the threshold. The formula for the blur detection is:

$$blurred = (V < T)$$

Where V is the variance of the Laplacian, and T is the threshold value. This formula returns a boolean value (true or false) depending on whether the image has a low or high variance of the Laplacian. You can use this formula to categorize images based on their blur levels.

4. Function for categorizing images as Night-Time image

function *categorize_nighttime*

This function uses the mean and standard deviation of pixel intensities to measure the darkness and contrast of an image. Here are steps for this function:

4.1 Convert the image to grayscale:

Please refer section 1.1.

4.2 Calculate the mean and standard deviation of pixel intensities

Please refer to section 1.5.1 and section 1.5.2 for mean and standard deviation calculations, respectively.

4.3 Thresholding

(1) The mean intensity of a nighttime image is usually low, because there is less light source at night. A possible condition is:

$$\mu < 70$$

(2) The standard deviation of a nighttime image is usually low, because there is less contrast and details at night. A possible condition is:

$$\sigma < 30$$

The final nighttime detection is the logical AND of these conditions, which means that both of them must be satisfied for the image to be classified as nighttime. A formula for this detection is:

$$nighttime = (\mu < 70) \Lambda (\sigma < 30)$$

Where μ is the mean intensity, σ is the standard deviation, and Λ the logical AND operator. User can adjust the threshold. This formula returns a boolean value (true or false) depending on whether the image meets the criteria for being nighttime or not.