Scania Task - Perception Testing

Nikhil Challa

October 5, 2023

1 Introduction

The task at hand is to classify the images with environment attributes namely fog, snow, rain and clear sky. I will treat this as classification problem, where the images are classified as any of the the attribute. Meaning, the algorithm will attempt to run the detection for all three types (fog, snow, rain) but only select the one that fits best. Clear sky is declared if none of the other attributes qualify. The code provided uses methods from image analysis to determine the attributes.

Since the task is to detect active snow and not just snow collected in the background, the existing labeled dataset images are not usable. Also most images are not accurate representation of images seen from AV, hence additional effort was made to select the right images. Apart from existing Kaggle datasets and from hrefhttps://scale.com/Scale AI, images were collected from other sources such as pexel and shutterstock. These images are free to use as long as the images are not monitized or distributed on other platforms. Please refrain from using these images beyond trying them for testing the source code.

The source code is written in MATLAB, with utilized built-in functions listed in the appropriate section.

To describe the source code, will cover a feature summary of each attribute and one example of matching and different attribute for clarification. From the information provided, and referring to [1], it appears Lidar suffers under bad weather conditions and additional precautions in object detection needs to be taken. The source code provided by myself depends on richness of the images to detect fog and snow with better certainty.

2 Fog Detection

Referring to [2], few properties of images with fog can be iterated. Color images tend to have lower color distribution (across R-G-B), and the intensity of each channel is on the lower intensity (hence more gray tones). Using the dehazing source code provided by the same reference author, we should see better color distribution and intensity. Unfortunately dehazing sometime creates artificial color tones, unless the image itself is know to be foggy as prior knowledge. Since this knowledge is not known, the dehazing process is applied to objects detected in the image. For detection objects, I use a built-in MATLAB function *yolov4ObjectDetector* derived from YOLO4 publication [3]. The defogging function fails for images of small sizes, hence I limit the bounding box to 50 pixels on either axes.

Figure 1: Example of fogging image with object detection

Once applying defogging to each object, an std of each pixel is determine. The mean and std (this is std of an std) of all the pixes are then plotted and compared against the object property priot to dehazing. If the object has sufficient color distribution, a clear increase in both the mean and std is objected.

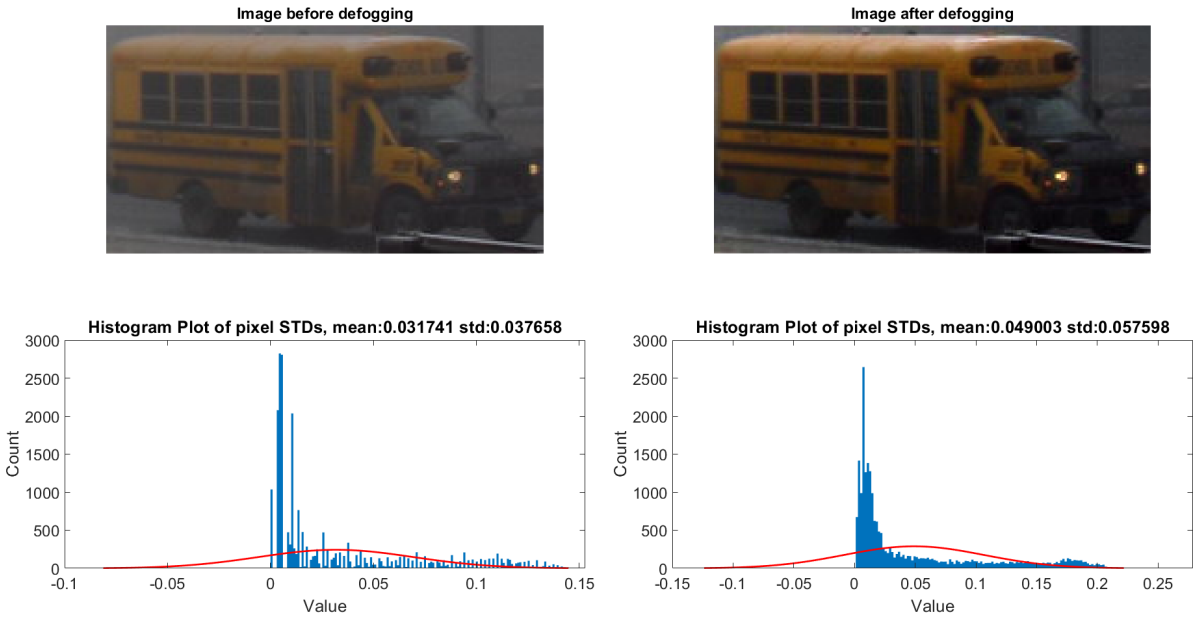

Figure 2: Foggy - Histogram plots for mean and std of the std of each pixel for Object 1

All objects detected in the image is plotted together. A vector is a line drawn from the point before dehazing (blue dot) to the point after dehazing (red dot) for every object. If there is more than one object recognized in the image, the vectors are added together and the mean is taken (divide the resulting vector by the number of objects). The absolute value of this vector is then used to determine the possibility of fogging. The method to determine the probability of each environment attribute has not been identified, and a simple thresholding is used.

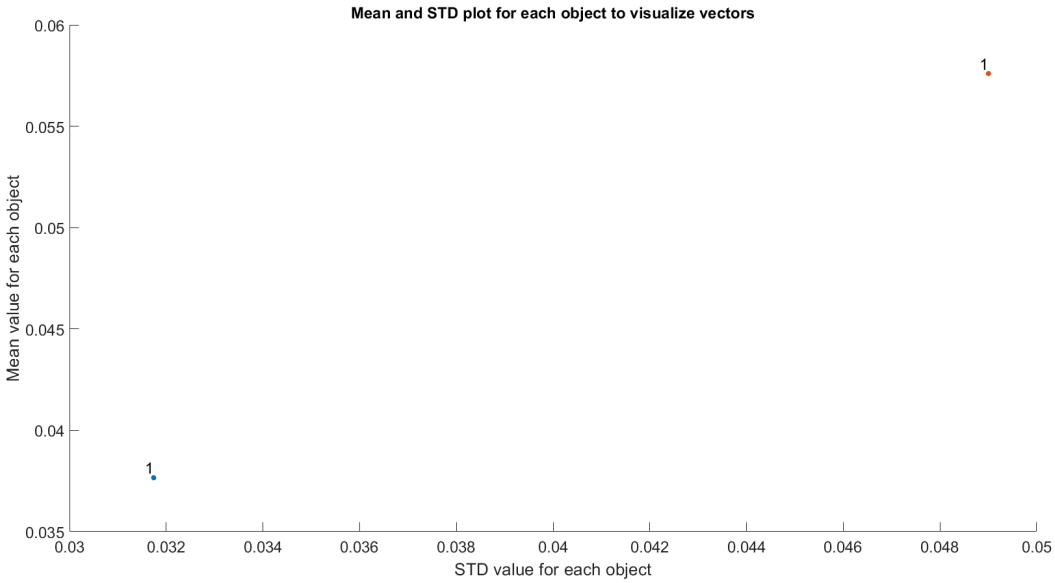

Figure 3: Foggy - Plotting mean and std for each object

The above steps will be repeated for an image that has no fog.

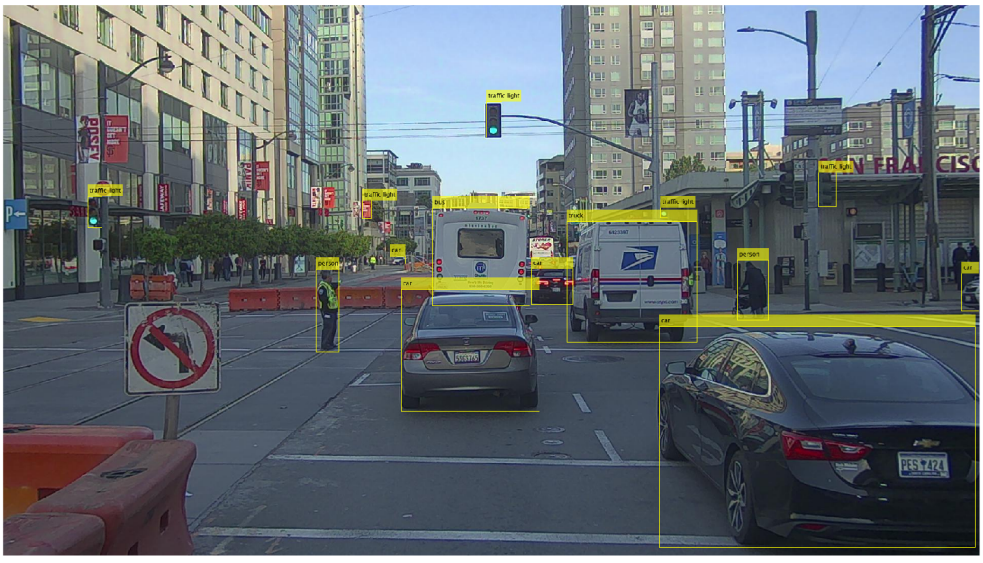

Figure 4: Example of no fogging image with object detection

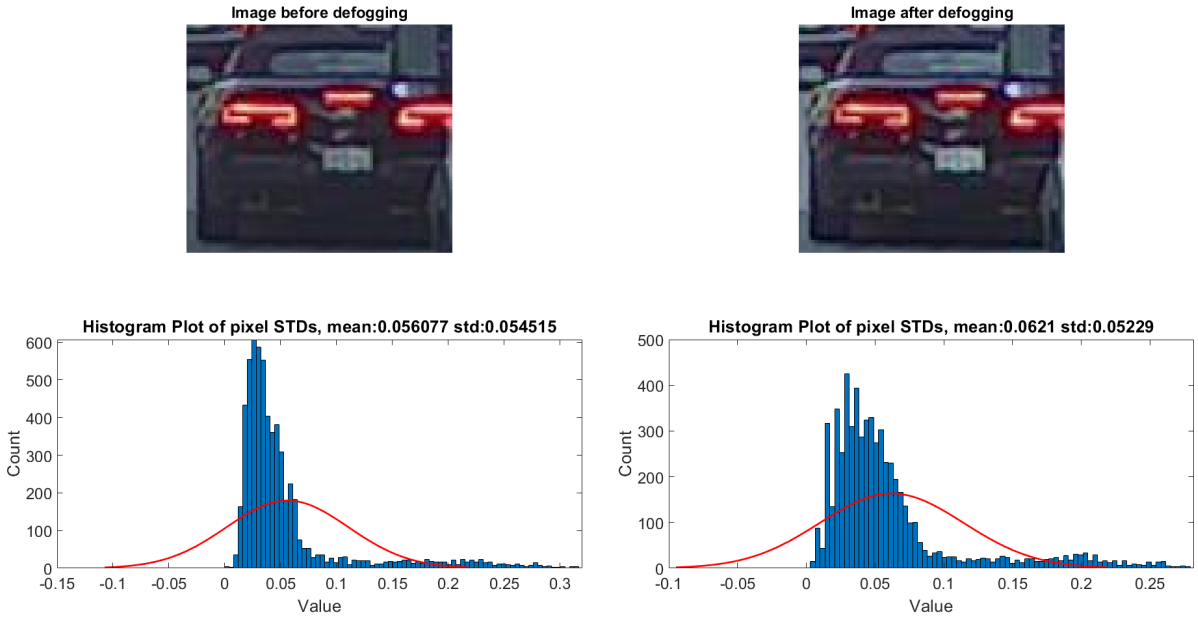

Figure 5: Not foggy - Histogram plots for mean and std of the std of each pixel for Object 1

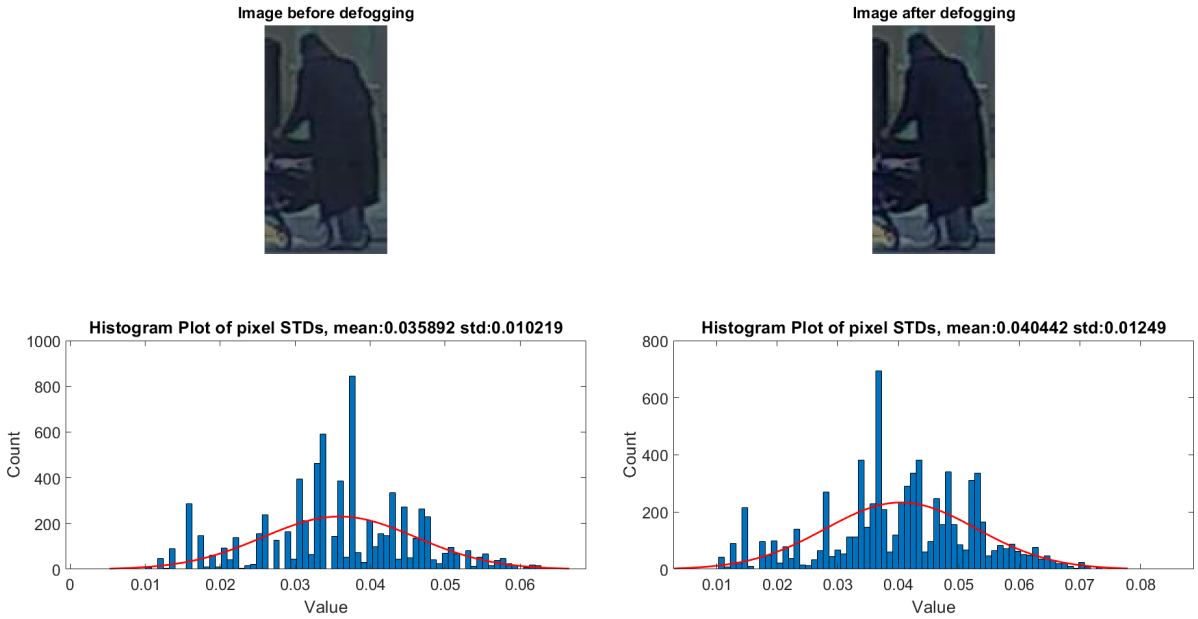

Figure 6: Not foggy - Histogram plots for mean and std of the std of each pixel for Object 2

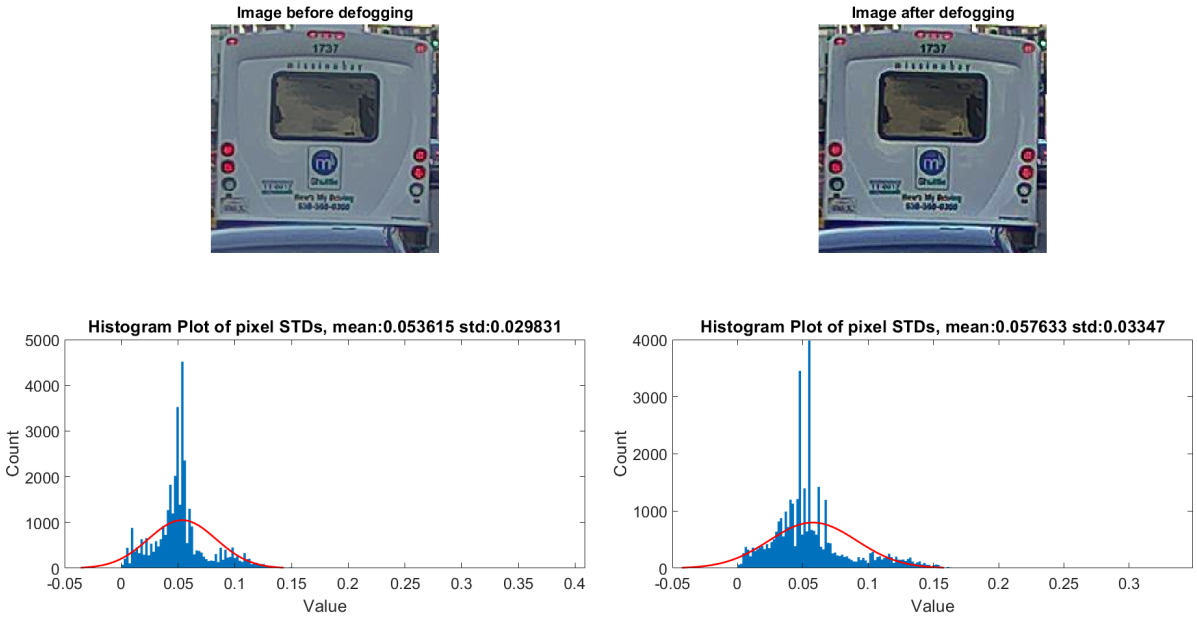

Figure 7: Not foggy - Histogram plots for mean and std of the std of each pixel for Object 2

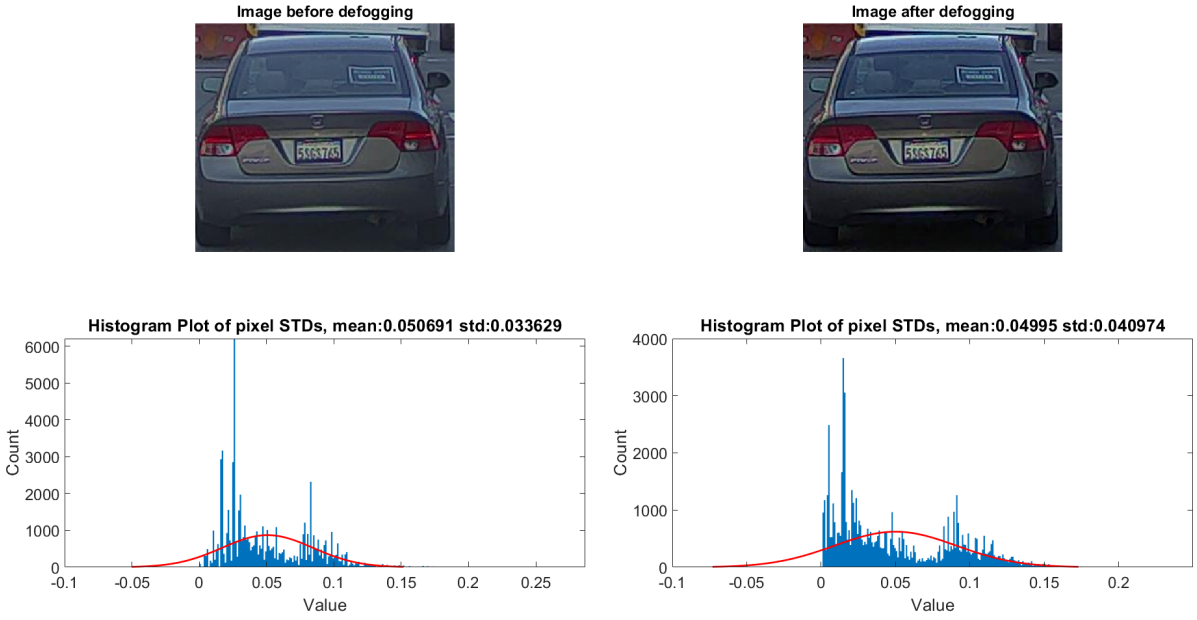

Figure 8: Not foggy - Histogram plots for mean and std of the std of each pixel for Object 2

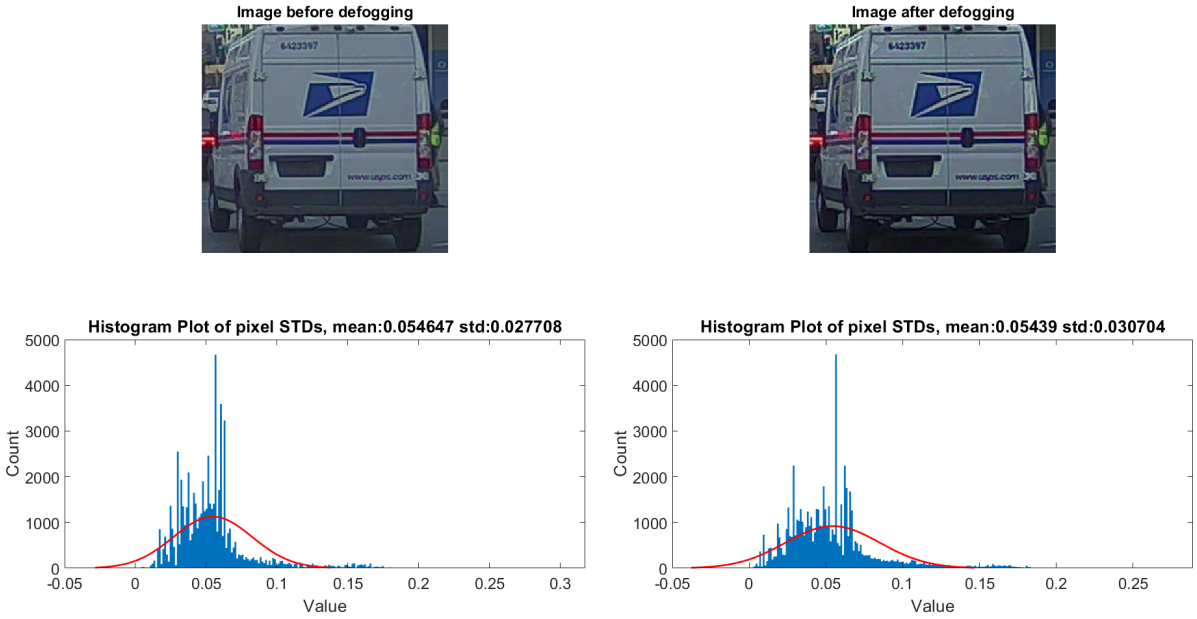

Figure 9: Not foggy - Histogram plots for mean and std of the std of each pixel for Object 2

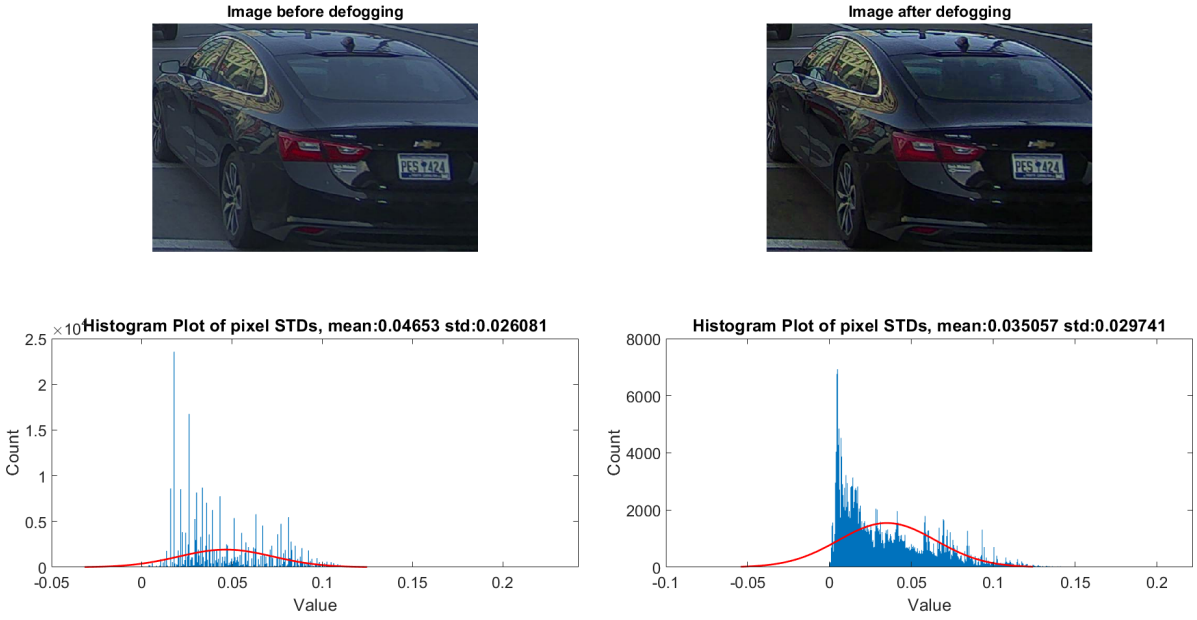

Figure 10: Not foggy - Histogram plots for mean and std of the std of each pixel for Object 2

Notice that the vectors dont all align in the +ve direction. For example, vector for object1, points almost horizontally but slighly in the -ve y-axis. Vector for object 3 points in the +ve direction for both axes. This randomness in the vector direction is due to attempt to defog an image that is not foggy. The defogging for this image does not always ensure increasing mean and std value. When vectors in somewhat random directions are accumulated the resulting vector is small. Taking the mean then reduces the vector intensity even further. One additional condition is added to the code, which is if the accumulated vector does not point in the +ve direction for both axes, it is set to 0.

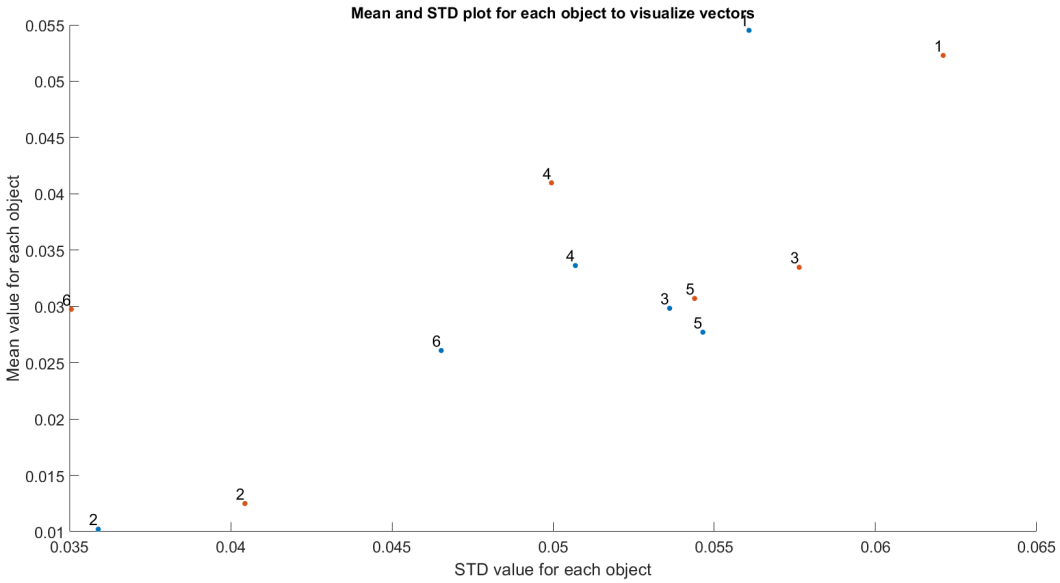

Figure 11: Not foggy - Plotting mean and std for each object

After evaluation with multiple images, a threshold value of 0.01 is selected. If the resulting vector magnitude is greater than 0.01, the image is classified as having fog.

3 Snow Detection

For snow detection, there were complexities with correctly classifying images that had snow in the background but no active snowing vs active snowing. For reliable detection, the image needs to contain objects. Again, the same YOLO4 method was utilized for this detection step. The following steps are performed for each of the boxed object images.

- Identify pixels that have an intensity greater than 128 for all three channels and the standard deviation value of the three channel (same pixel) is less than 20
- Use *bwareafilt* to remove pixels whose connected shape exceeds the predetermine size. This is to discard background white images (snow collected on the ground or on trees) or obejcts that are near shades of white.
- Count remaining pixels in the boxed images for calculation percentage of snow particles in the boxed image ($TotalCount$).
- Perform image closing using *imclose* to discard streaks that are too close to each other. These streaks are usually from the background and not really contributing to actual snow particles.
- Count the remaining pixels that are marked as snow particles ($NumSnowPixels$).
- Calculate ratio of ($NumSnowPixels$) to ($TotalCount$).
- If more than one object is detected in the image, the mean values of all the ratios can be taken as the final value.

Figure 12: Example of an obejct detected image with active snowfall

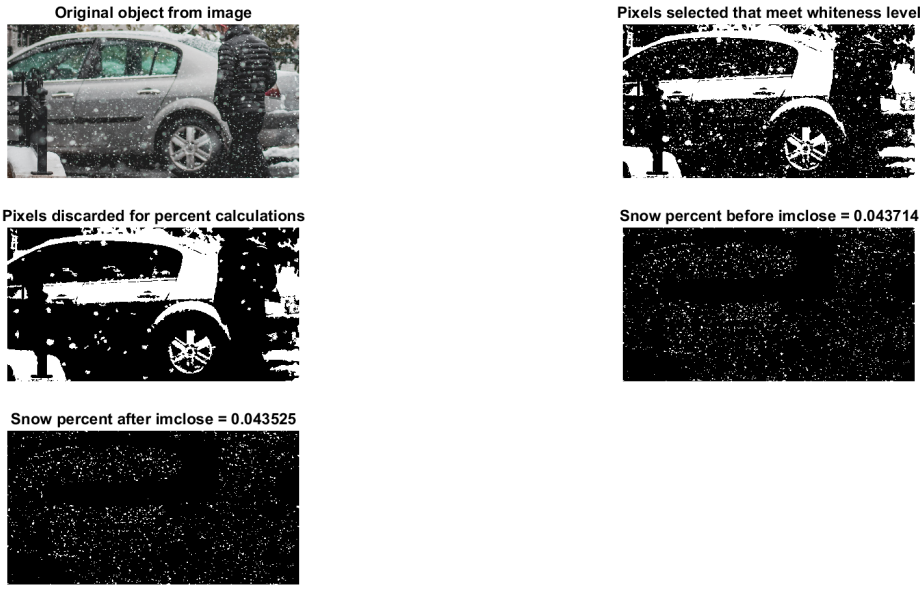

Figure 13: Snow - Object 1 with percentage calculations

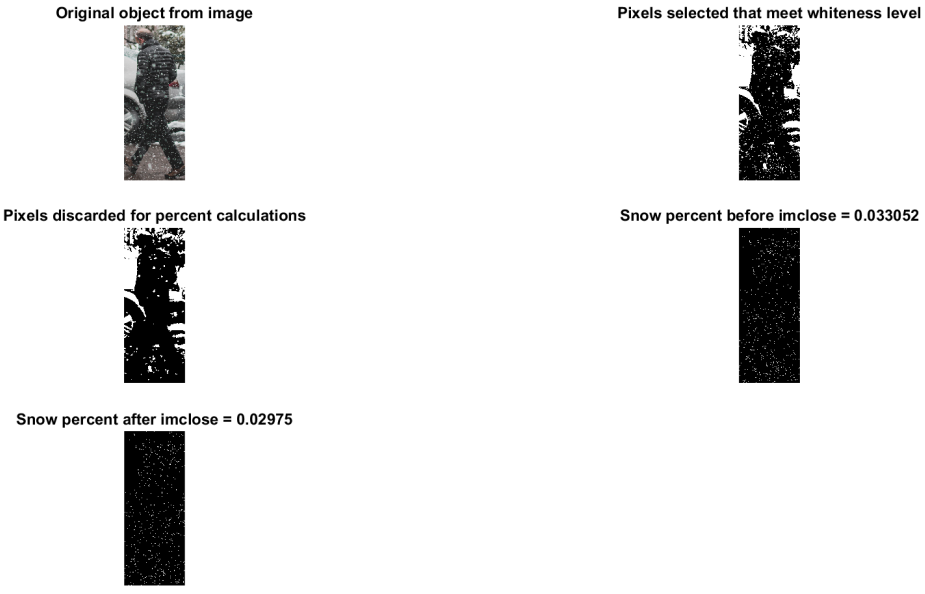

Figure 14: Snow - Object 2 with percentage calculations

The same process is repeated for an image that dont show active snowing.

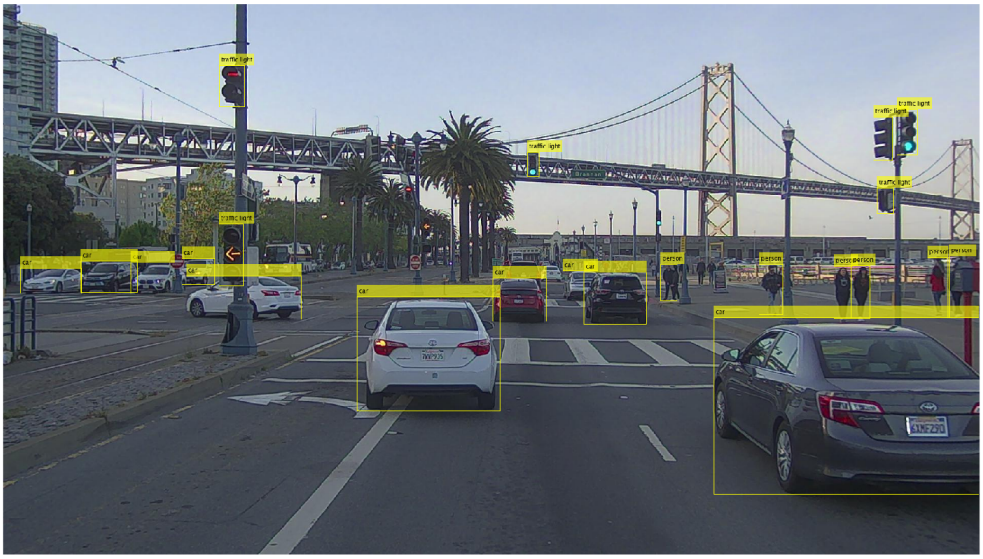

Figure 15: Example of an obejct detected image with no active snowfall

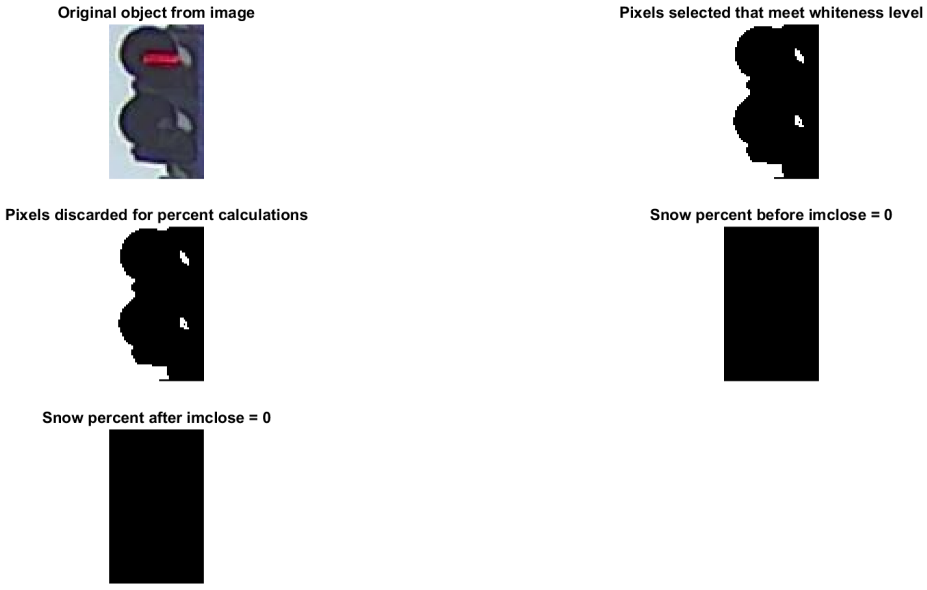

Figure 16: No snow - Object 1 with percentage calculations

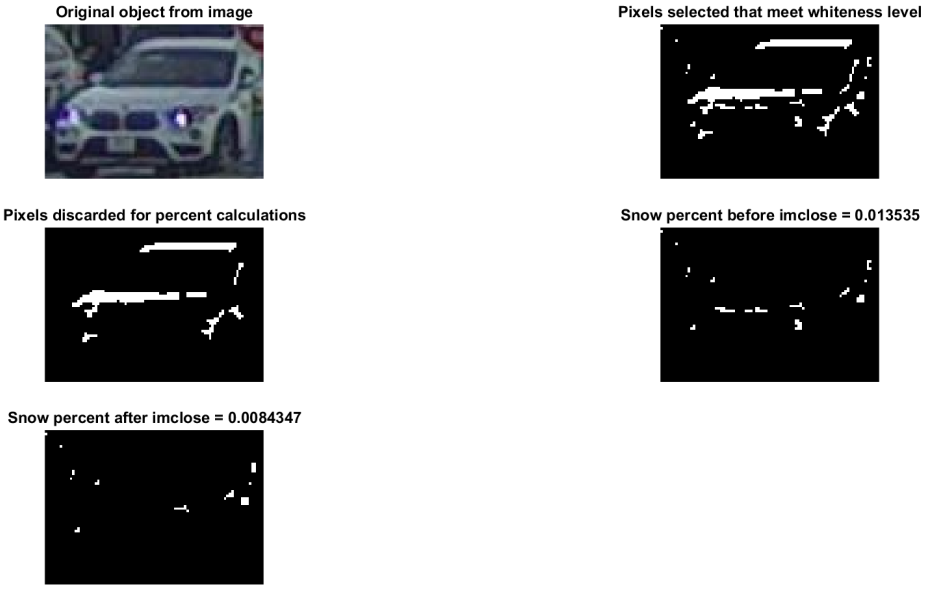

Figure 17: No snow - Object 2 with percentage calculations

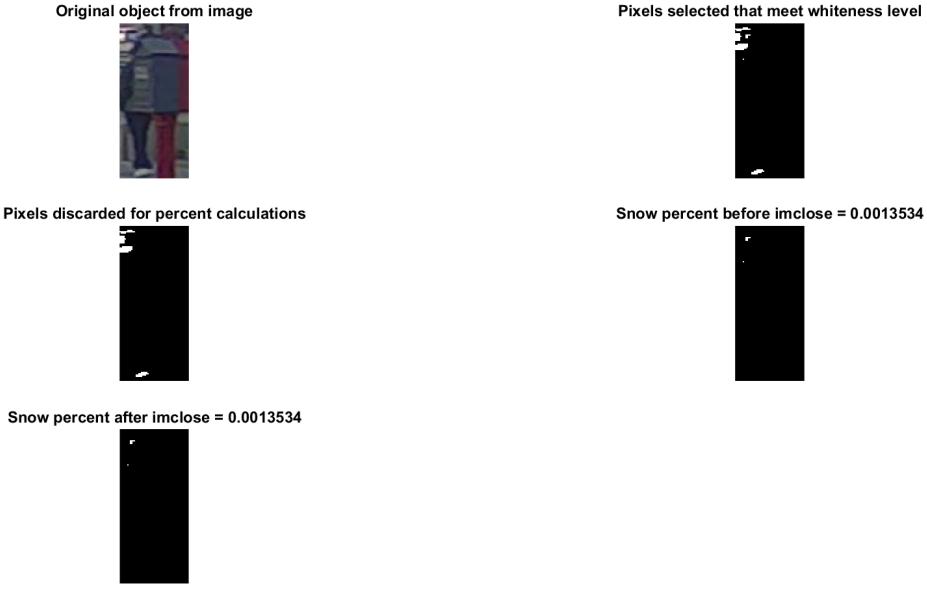

Figure 18: No snow - Object 3 with percentage calculations

After evaluation with multiple images, a threshold value of 0.01 is selected. If the resulting percentage of pixel is greater than 1%, the image is classified as having active snowfall.

4 Rain Detection

Referring to [4], properties of rain droplets on the image can be used. To iterate, rain induces gradients in the images, and has high reflective index when hitting the surface. The high reflective index is due to light reflecting off the surface that contains water, and the ripples due to water hitting the surface also creates gradients. For rain detection, the entire image is utilized.

Few additional properties were observed. The whiteness of ripples hitting surface was higher than rain in mid air. For this two different methods were used,

- Select pixels that meet certain grayness level (each channel value between 100 and 200, and the standard deviation less than 60).
- Select pixels that meet certain gradient intensity (greater than or equal to 50).

Using an aggregate of pixels with the above two methods, a more descriptive representation of rain is covered. To avoid falsely detected gradients due to natural color imbalances such as dirt on the road, grass, and trees, the spatial distribution of the selected pixels is evaluated.

Figure 19: Example of an image with active rain fall

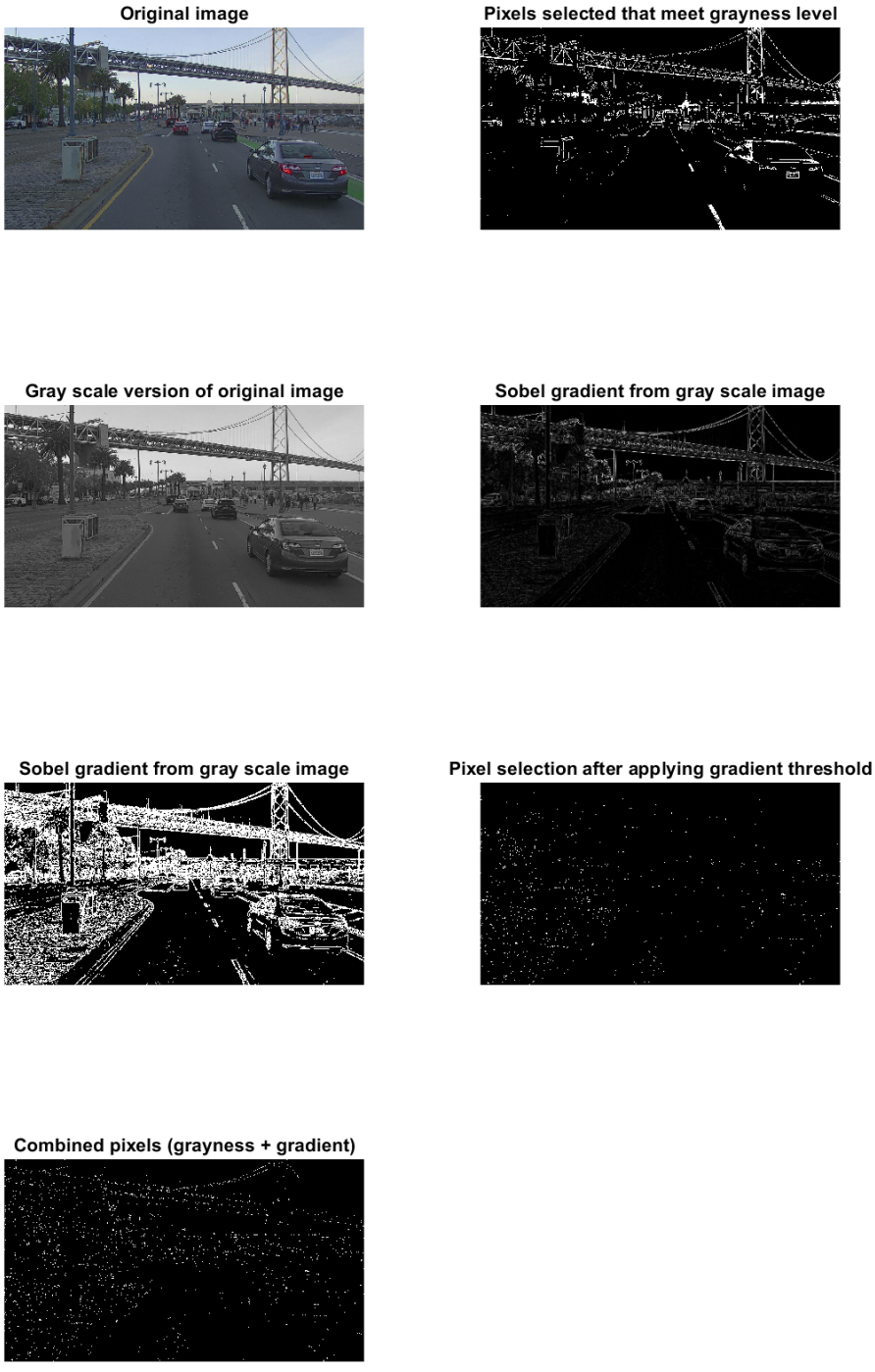

Figure 20: Example of an image with active rain fall

After evaluation with multiple images, a minimum threshold value 0.01 in selected pixel percentage from entire image, and the standard deviation less than 1000 in the number of pixels in each subsection is selected.

5 Source Code Details

For simplicity, there is only one file **ClimateClassification.m** to be utilized to test the classification performance. The input or the location to the image file can be provided at the very top of the script. A lot of examples are provided as comments, but you are free to provide your own image. The results will be displayed in the command window.

labelpoints.m was picked from labelpoints and the contents within folder *Dark-Channel-Haze-Removal* are picked from Single Image Haze Removal Using Dark Channel Prior. The source for images in the other fodlers are already covered in the introduction section

6 Conclusion and Summary on the Results

The fog detection algorithm is the least reliable one, while the snow detection algorithm is the most reliable. In the even that more than one attribute is selected, the following hierarcy is followed...

- If snow attributed is selected, the final conclusion will indicate *## Final Conclusion : This image has snowfall ##*
- If rain attributed is selected, the final conclusion will indicate *## Final Conclusion : This image has rainfall ##*
- If fog attributed is selected, the final conclusion will indicate *## Final Conclusion : This image has fog ##*
- If none of the three attributes (rain, snow, fog) are selected, the final conclusion will indicate *## Final Conclusion : This image has clear weather ##*
- In the event of lack of objects detected in the image, an additional image *Uncertainity in Fog and Snow* will be displayed

References

[1] T. Vattem, G. Sebastian, and L. Lukic, "Rethinking lidar object detection in adverse weather conditions," in *2022 International Conference on Robotics and Automation (ICRA)*, pp. 5093–5099, 2022.

[2] K. He, J. Sun, and X. Tang, "Single image haze removal using dark channel prior," in *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1956–1963, 2009.

[3] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection," 2020.

[4] H. Wang, Y. Wu, M. Li, Q. Zhao, and D. Meng, "Survey on rain removal from videos or a single image," *Science China Information Sciences*, vol. 65, dec 2021.