RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	COE843
Course Title	Introduction to Computer Vision
Semester/Year	F2023

Instructor	Guanghui Richard Wang

ASSIGNMENT No.	2

Assignment Title	Problem Set #2

Submission Date	October 22th, 2023
Due Date	October 23th, 2023

Student Name	Hamza Iqbal
Student ID	500973673
Signature*	H.I

**By signing above you attest that you have contributed to this written lab report and confirm that all work you have swung the lab contributed to this lab report is your own work.*

Part 1

Question 1:

```
%% Problem 1.

% Display Original Image
img = imread("lion.jpg");
imshow(img);
title ('The Lion - Original Image');

% Display Grey Image
subplot(2, 2, 1);
imggray = rgb2gray(img);
imshow(imggray);
title ('The Lion - Grayscale Image');

% Robert Edge Detector
subplot(2, 2, 2);
robert = edge(imggray,"roberts");
imshow(robert);
title('Robert Mask');

% Prewitt Edge Detector
subplot(2, 2, 3);
prewitt = edge(imggray,"prewitt");
imshow(prewitt);
title('Prewitt Mask');

% Sobel Edge Detector
subplot(2, 2, 4);
sobel = edge(imggray,"sobel");
imshow(sobel);
title('Sobel Mask');
```

Code 1: Code for the Robert, Prewitt, and Sobel masks respectively

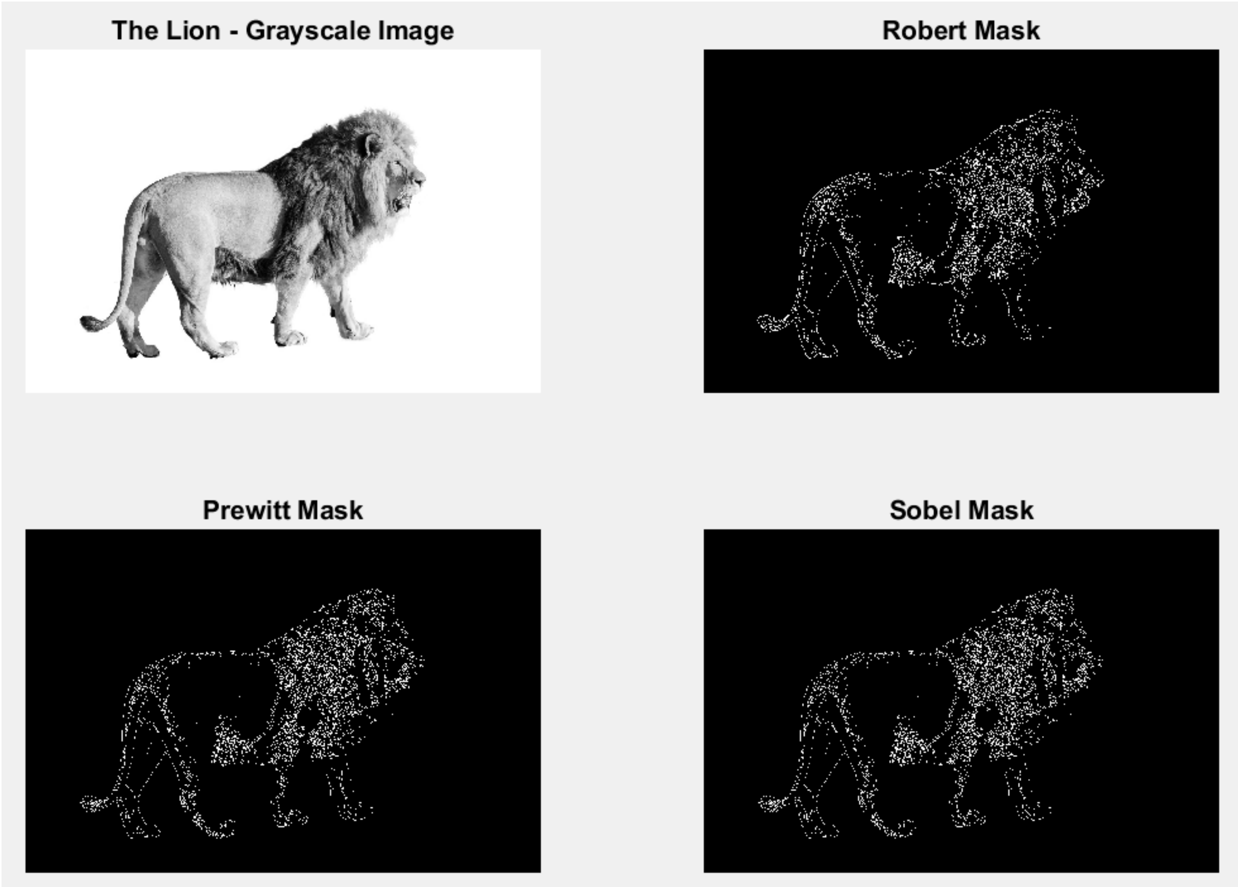

Figure 1: Image Results for the Greyscale, Robert, Prewitt, and Sobel masks respectively on a grayscale image of a lion

All of the edge detectors outline the edges of the lion in the images. It seems the Robert mask is good at detailing diagonal edges. Both horizontal and vertical edges are detected by Prewitt and Sobel's filters. However it seems that the Prewitt & Sobel ignore the pixels in the center, whereas the Robert mask doesn't. Overall, it seems the Robert filter produces slightly weaker outlines in contrast to the Sobel & Prewitt filter. The Sobel detector appears to be the most detailed edge detector of the three.

<u>Question 2:</u>

Based on the image, the f (x) is:

$f\ (x)$: 6,6,6,6,5,4,3,2,1,1,1,1,1,1,6,6,6,6,6 0

0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9

$f'(0)$:

The formula for the first-order derivative of the function f(x) is:

$$f'(x) = f(x + 1) - f(x)$$

Computing the first-order derivative of f(x) gives the results:

$f'(0) = f(2) - f(1) = 6 - 6 = 0$

$f'(1) = f(3) - f(2) = 6 - 6 = 0$

$f'(2) = f(4) - f(3) = 6 - 6 = 0$

$f'(3) = f(5) - f(4) = 5 - 6 = -1$

$f'(4) = f(6) - f(5) = 4 - 5 = -1$

$f'(5) = f(7) - f(6) = 3 - 4 = -1$

$f'(6) = f(8) - f(7) = 2 - 3 = -1$

$f'(7) = f(9) - f(8) = 1 - 2 = -1$

$f'(8) = f(10) - f(9) = 1 - 1 = 0$

$f'(9) = f(11) - f(10) = 1 - 1 = 0$

$f'(10) = f(12) - f(11) = 1 - 1 = 0$

$f'(11) = f(13) - f(12) = 1 - 1 = 0$

$f'(12) = f(14) - f(13) = 6 - 1 = 5$

$f'(13) = f(15) - f(14) = 6 - 6 = 0$

$f'(14) = f(16) - f(15) = 6 - 6 = 0$

$f'(15) = f(17) - f(16) = 6 - 6 = 0$

$f'(16) = f(18) - f(17) = 6 - 6 = 0$

$f'(17) = f(19) - f(18) = 6 - 6 = 0$

$f'(18) = f(19) - f(18) = 0 - 6 = -6$

Calculating the 2nd-order derivative gives:

The formula for the 2nd-order derivative of the function f(x) is:

$f''(x) = f(x - 1) + f(x + 1) - 2f(x)$

Computing the 2nd-order derivative of f(x) gives the results:

$f''(0) = f(-1) + f(1) - 2f(0) = 0 + 6 - 12 = -6$
$f''(1) = f(0) + f(2) - 2f(1) = 6 + 6 - 12 = 0$
$f''(2) = f(1) + f(3) - 2f(2) = 6 + 6 - 12 = 0$
$f''(3) = f(2) + f(4) - 2f(3) = 6 + 5 - 12 = -1$
$f''(4) = f(3) + f(5) - 2f(4) = 6 + 4 - 10 = 0$
$f''(5) = f(4) + f(6) - 2f(5) = 5 + 3 - 8 = 0$
$f''(6) = f(5) + f(7) - 2f(6) = 4 + 2 - 6 = -0$
$f''(7) = f(6) + f(8) - 2f(7) = 3 + 1 - 4 = 0$
$f''(8) = f(7) + f(9) - 2f(8) = 2 + 1 - 2 = 1$
$f''(9) = f(8) + f(10) - 2f(9) = 1 + 1 - 2 = 0$
$f''(10) = f(9) + f(11) - 2f(10) = 1 + 1 - 2 = 0$
$f''(11) = f(10) + f(12) - 2f(11) = 1 + 1 - 2 = 0$
$f''(12) = f(11) + f(13) - 2f(12) = 1 + 1 - 2 = 0$
$f''(13) = f(12) + f(14) - 2f(13) = 1 + 6 - 2 = 5$
$f''(14) = f(13) + f(15) - 2f(14) = 1 + 6 - 12 = -5$
$f''(15) = f(14) + f(16) - 2f(15) = 6 + 6 - 12 = 0$
$f''(16) = f(15) + f(17) - 2f(16) = 6 + 6 - 12 = 0$
$f''(17) = f(16) + f(18) - 2f(17) = 6 + 6 - 12 = -0$
$f''(18) = f(17) + f(19) - 2f(18) = 6 + 0 - 12 = -6$

<u>Question 3:</u>

```matlab
%% Problem 3.

img = imread("lion.jpg");

% Greyscale Image
subplot(2, 2, 1);
imggray = rgb2gray(img);
imshow(imggray);
title('The Lion - Grayscale Image');

% Sharpened Image
sharpen = imsharpen(imggray);
subplot(2, 2, 2);
imshow(sharpen);
title('The Lion - Sharpened Image');

% High Boost Filtered Image k=1
subplot(2, 2, 3);
hbf = imsharpen(imggray, "radius", 5, "amount", 1);
imshow(hbf);
title('HBF Lion Image k=1');


% High Boost Filtered Image k=5
subplot(2, 2, 4);
hbf2 = imsharpen(imggray, "radius", 5, "amount", 5);
imshow(hbf2);
title('HBF Lion Image k=5');
```

Code 2: Matlab code demonstrating the unmasking and high booster filter effect on a grayscale image via the Matlab built-in function imsharpen().

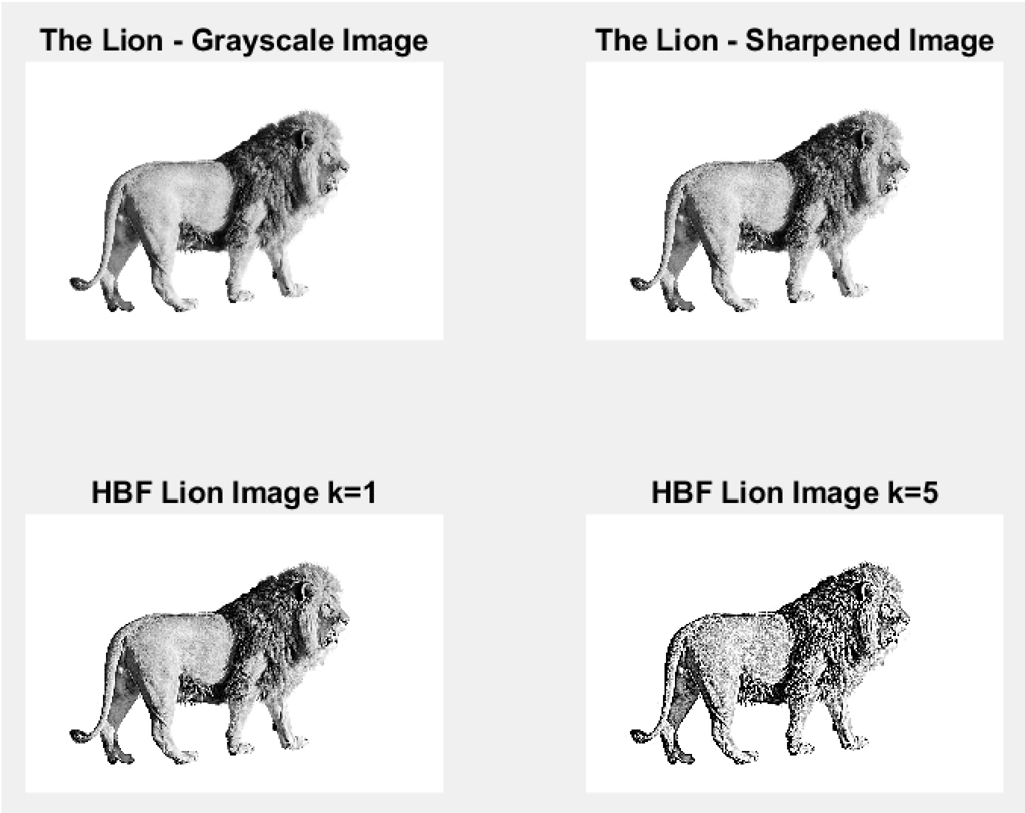

Figure 2: Image Results for the Greyscale, Sharpened, & high booster filter effect (k=1) & (k=5) on an image.

By first converting the original image to grayscale image, Matlab's built-in imsharpen() method was used to illustrate unsharp masking. In order to obtain a high pass filtered image, the image is essentially subtracted from a low pass filtered version image of itself, allowing for the image's high-frequency content to be preserved while its low-frequency content diminished using the high pass filter. Additionally, in order to sharpen the original image, the unsharp mask locates the edges of the various tones and boosts contrast. Therefore, while the original image has been altered, the appearance of the updated version is much sharper while maintaining the image's content.

Using the "radius" and "amount" parameters, unsharp masking enhances image edges and fine details by removing a blurry version of the image. Using high-boost filtering, the degree of improvement depends on the k value. A comparison of k=1 and k=5 shows the effect of sharpening intensity on photos. Although a higher k values adds more noise, it also improves edges.

Question 4:

```matlab
%% Problem 4.

% Grayscale Image
img = imread("lion.jpg");
imshow(img);
title('The Lion - Original Image');
imggray = rgb2gray(img);

% Noise Levels
noise1 = 0.25;
noise2 = 0.001;

% Gaussian noise #1
subplot(2,2,1);
gaus1 = imnoise(imggray,"gaussian", noise1);
imshow(gaus1);
title('Grayscale Image With Gaussian Noise of 0.25');

% Gaussian noise #2
subplot(2,2,2);
gaus2 = imnoise(imggray,"gaussian", noise2);
imshow(gaus2);
title('Grayscale Image With Gaussian Noise of 0.001');

% Average Filter
subplot(2,2,3);
AvgFilter = filter2(fspecial('average',3),gaus1)/255;
imshow(AvgFilter);
title('Average filter Gaus #1');

% Gaussian Filter
subplot(2,2,4);
GausFilter = imgaussfilt(gaus2, 0.9);
imshow(SecondGaus);
title('Gaussian filter on Gaus #2');
```

Code 3: Matlab code applying the Gaussian noise and Average, Gaussian Filters to Images, respectively

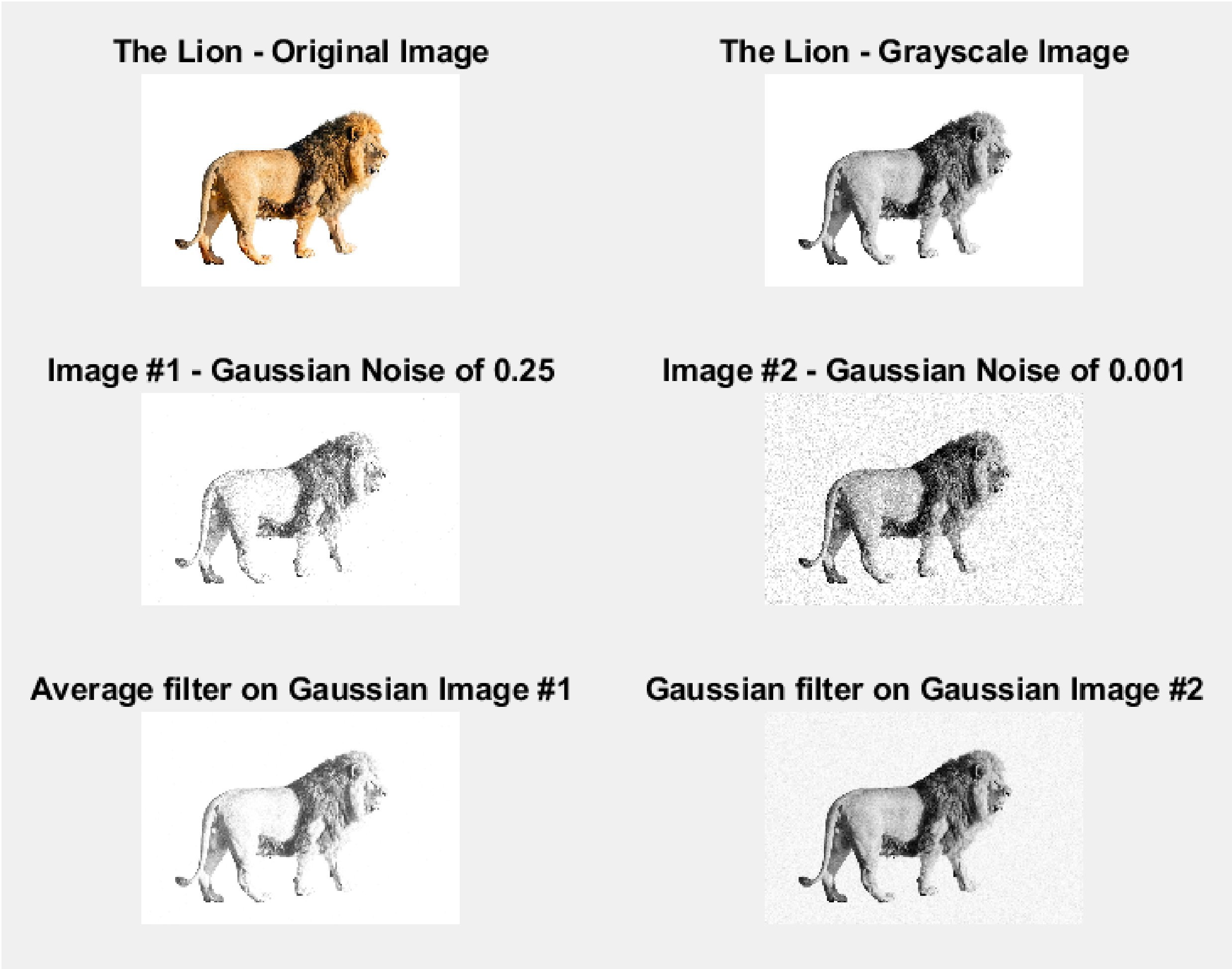

Figure 3: Images of Lion as the Original, in Grayscale, with noise applied, and the use of Average & Gaussian filters

Gaussian noise is applied to the grayscale with a value of 0.25, as shown in image #1 of Figure 3 above, which causes the image to lose contrast and adds noise. The larger value of 0.25 creates less contrast in comparison to image #2 on the right with a value of 0.001, while seemingly adding more noise than image #1, but less contrast. A specific filter can be used to restore these photos, which have lost their original quality. The Average filter, which averages the values of all the pixels around the neighboring pixels, causes the image to come out with a more clear quality. the Gaussian filter also causes the image to look smoother after the Gaussian filter is applied since less noise and low contrast are present.

Question 5:

1.

Formulas of 1st order derivative:

$$\frac{\partial f(x,y)}{\partial x} = \frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x) \qquad (1)$$

$$\frac{\partial f(x,y)}{\partial y} = \frac{\partial f}{\partial y} = f'(y) = f(y+1) - f(y) \qquad (2)$$

Formula for laplacian:

$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \qquad (3)$$

2.

For the given 3-bit image:

0 2 5 7
2 5 7 3
5 6 3 1
5 2 1 0

Sample calculation for the pixel (2,2), using equation (2) gives:

For $\frac{\partial f}{\partial x}$:

$\frac{\partial f}{\partial x} = f'(x) = f(x+1) - f(x)$
= f(2+1,2) - f(2,2)
= f(3,2) - (5)
= 3 - 5 = -2

For $\frac{\partial f}{\partial y}$:

$\frac{\partial f}{\partial y} = f'(y) = f(y+1) - f(y)$

= f(2+1,2) - f(2,2)
= f(3,2) - (5)
= 7 - 5 =2

The overall matrix for the x direction is:

2 3 2 -4
3 -2 -1 -2
1 -1 -2 -1
0 -3 -1 0

Similarly, the overall matrix for the y direction is:

2 3 2 2
3 1 -1 -2
1 -1 2 -2
-3 -3 -2 -1

3.

The laplacian mask given is:

0 1 0
1 -4 1
0 1 0

Computing the laplacian using equation (3) then results in:

2 -15 5 4
1 1 0 -7
4 -6 -2 -6
1 -5 1 0

4.

The laplacian can be normalized by using the max (max = 5) and min (min = -15) values, and then scaling it to a range of 0-255, hence centering it to 128.

Part 2

Using MATLAB's imcomplement () & reducehaze () functions, changes can be made to the original lowlight image to improve visibility and reduce haze effects.

```
%% Part 2

% Display Original Image
figure;
img = imread('lowlight.jpg');
imshow(img);
title ('Lowlight - Original Image');

% Display Greyscale Image
figure;
imggray = rgb2gray(img);
imshow(imggray);
title ('Lowlight - Grayscale Image');

% Inverted Image
figure;
imgInv = imcomplement(img);
imshow(imgInv);
title('Lowlight - Inverted Image');

% Apply haze reduction algorithm
figure;
haze = imreducehaze(imgInv);
imshow(haze);
title('Lowlight - Haze Reduction on Inverted Image');

%  Invert the haze reduction
figure;
invHaze = imcomplement(haze);
imshow(invHaze);
title('Lowlight - Inverted Haze-Reduction Image');
```

Code 4: Matlab code inverting the image, applying the haze reduction, then inverting the haze reduced image

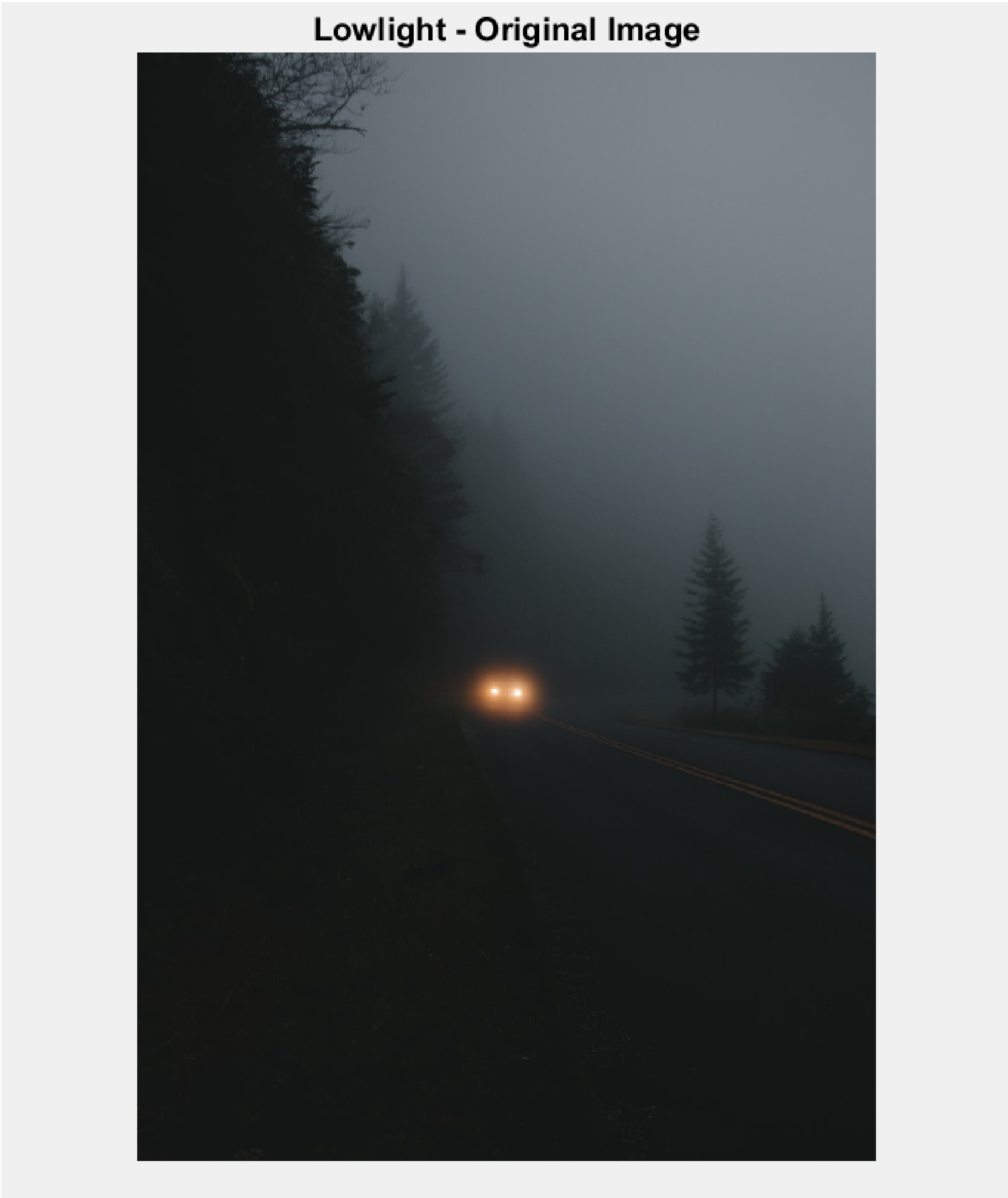

Figure 4: Lowlight Original Image

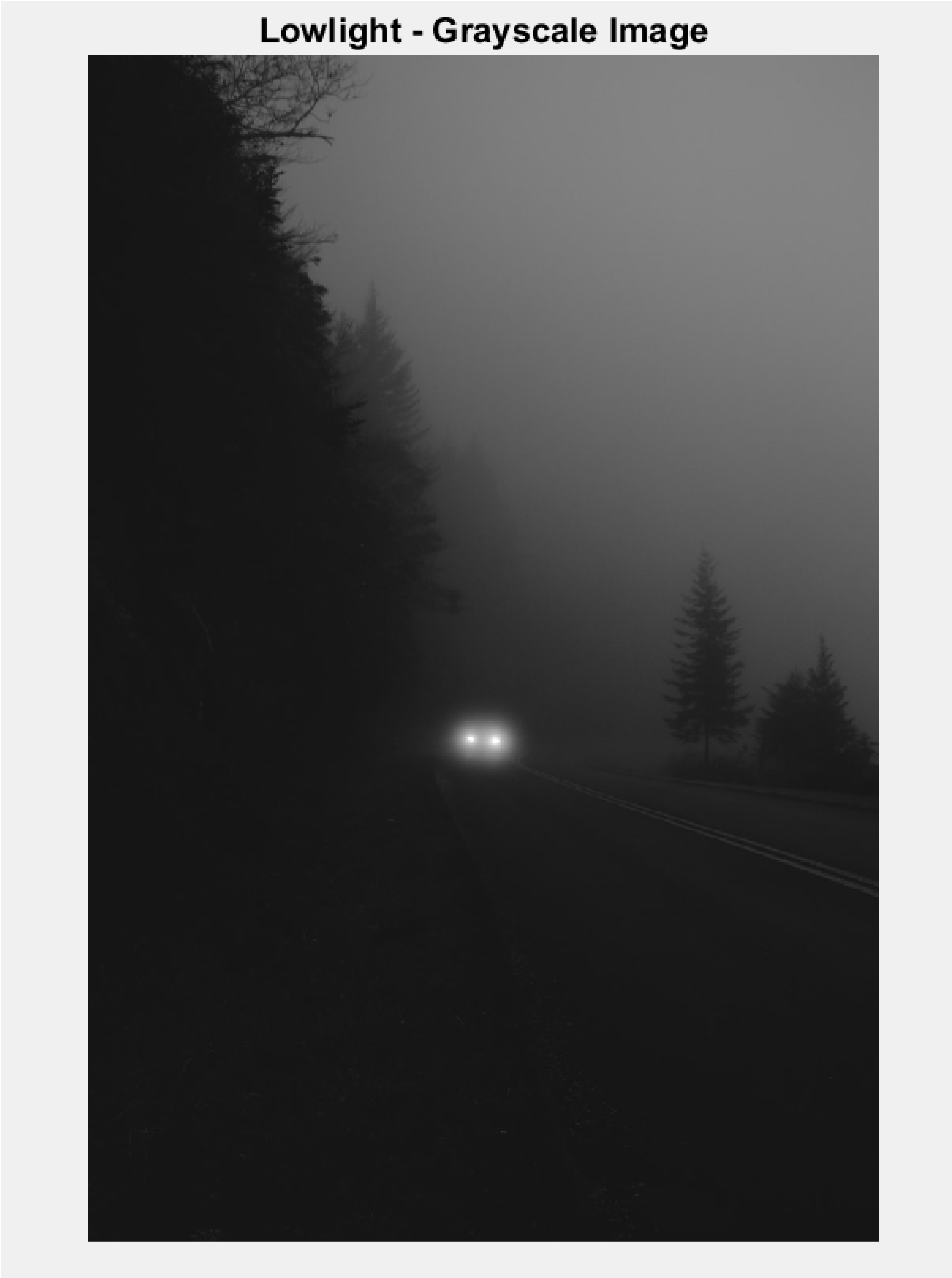

Figure 5: Lowlight Grayscale Image

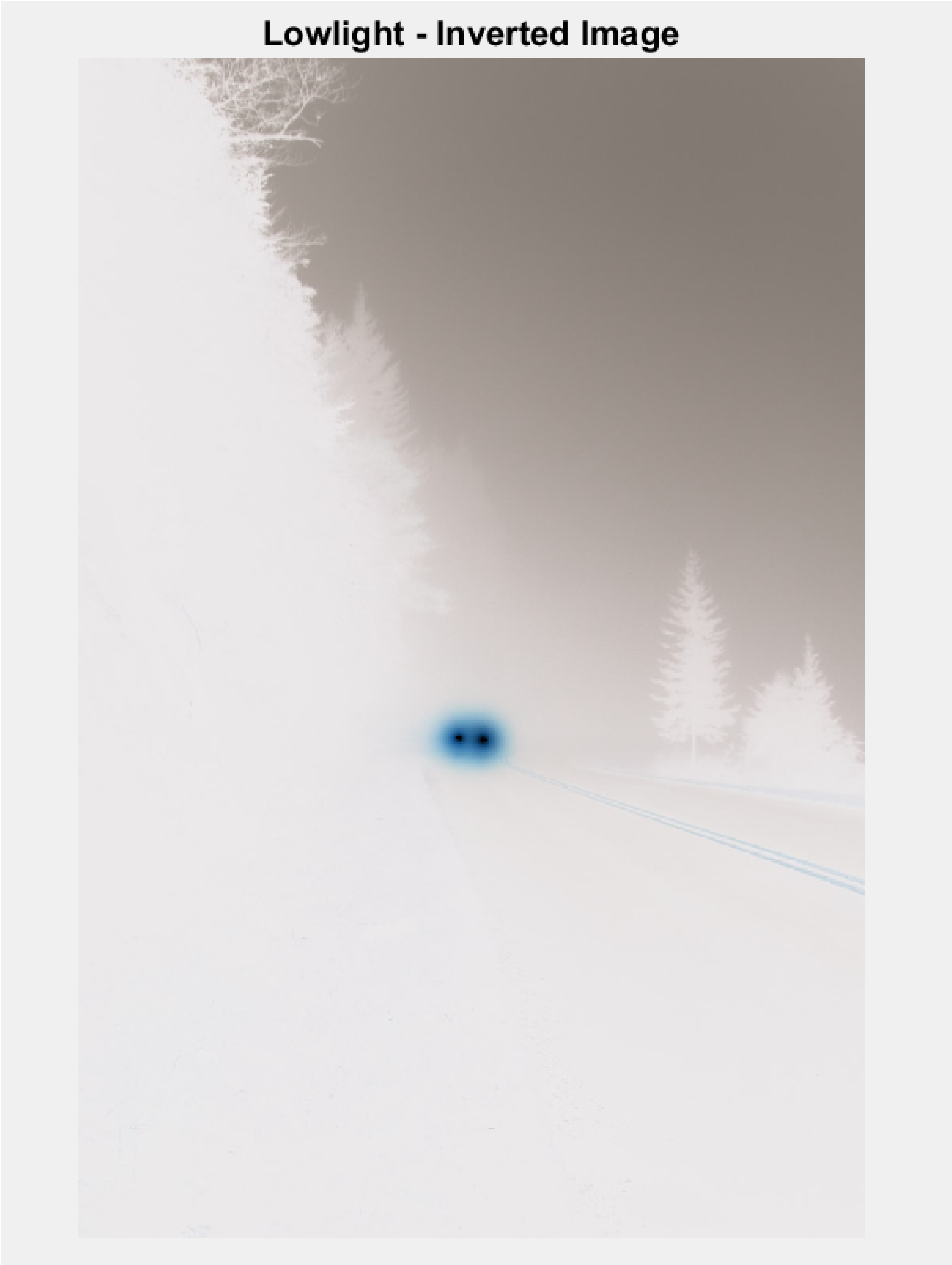

Figure 6: Lowlight Inverted Image

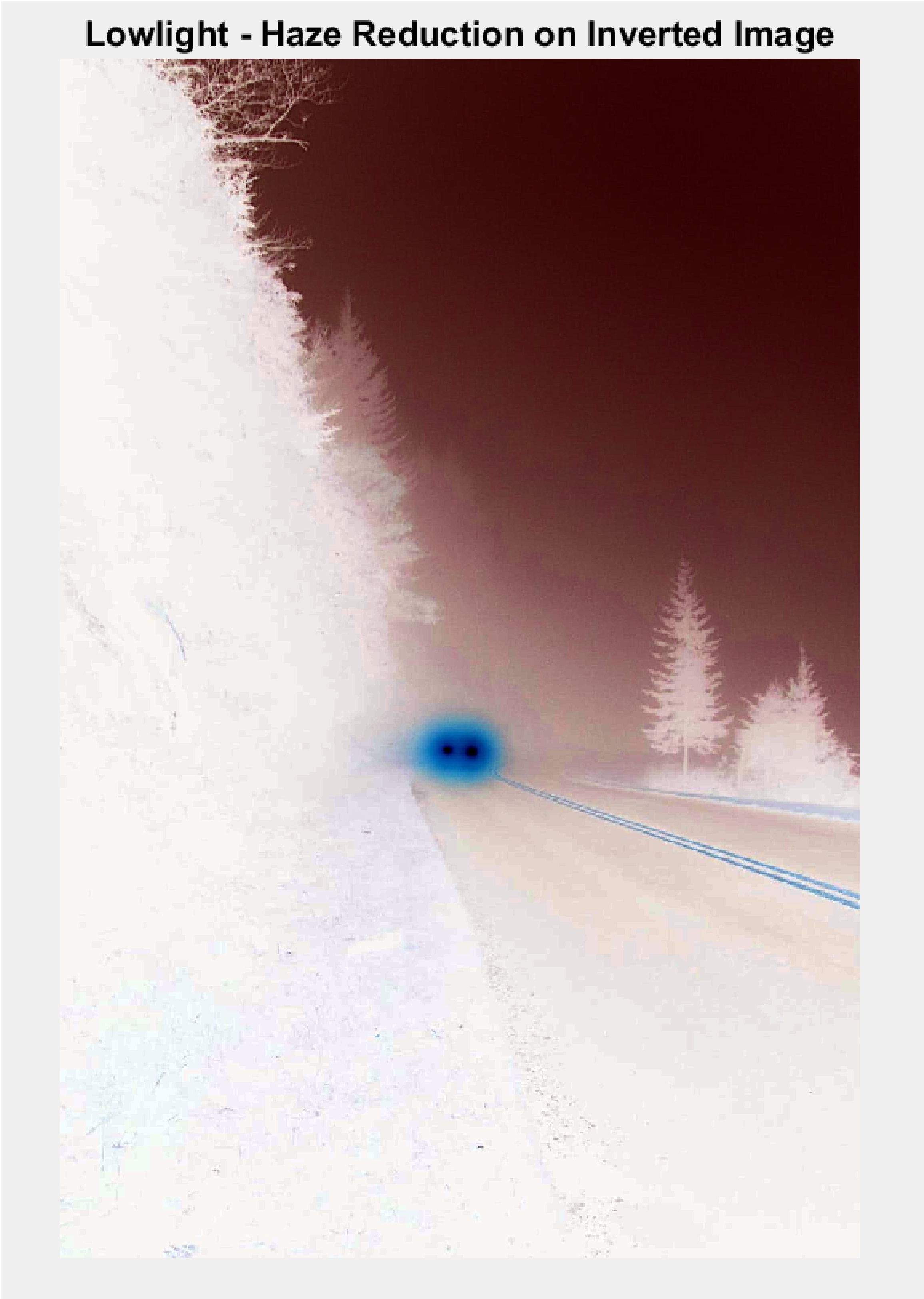

Figure 7: Haze-Reduced Effect on Inverted Lowlight Image

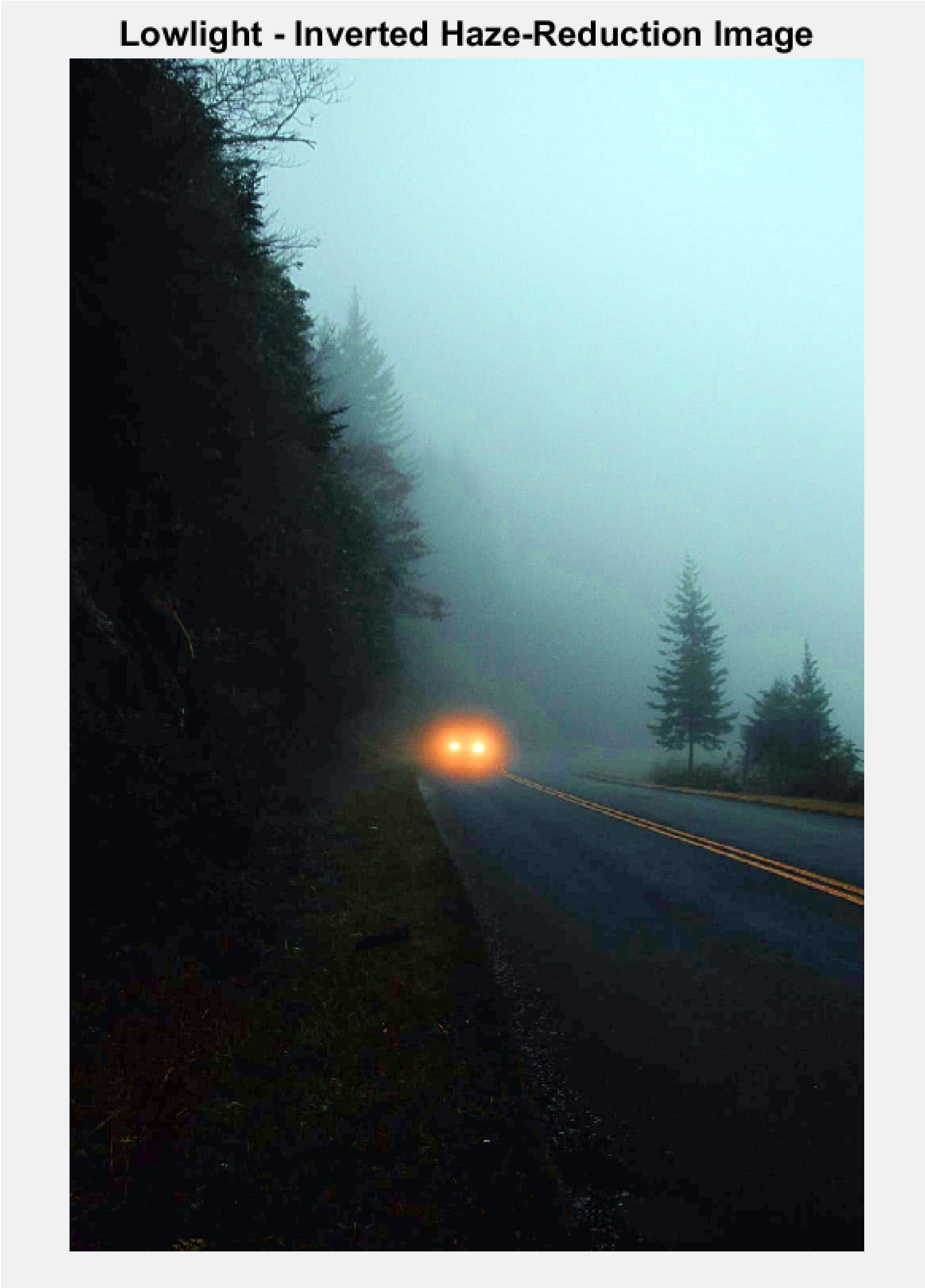

Figure 7: 'Inverted Haze-Reduced Effect' on Inverted Lowlight Image

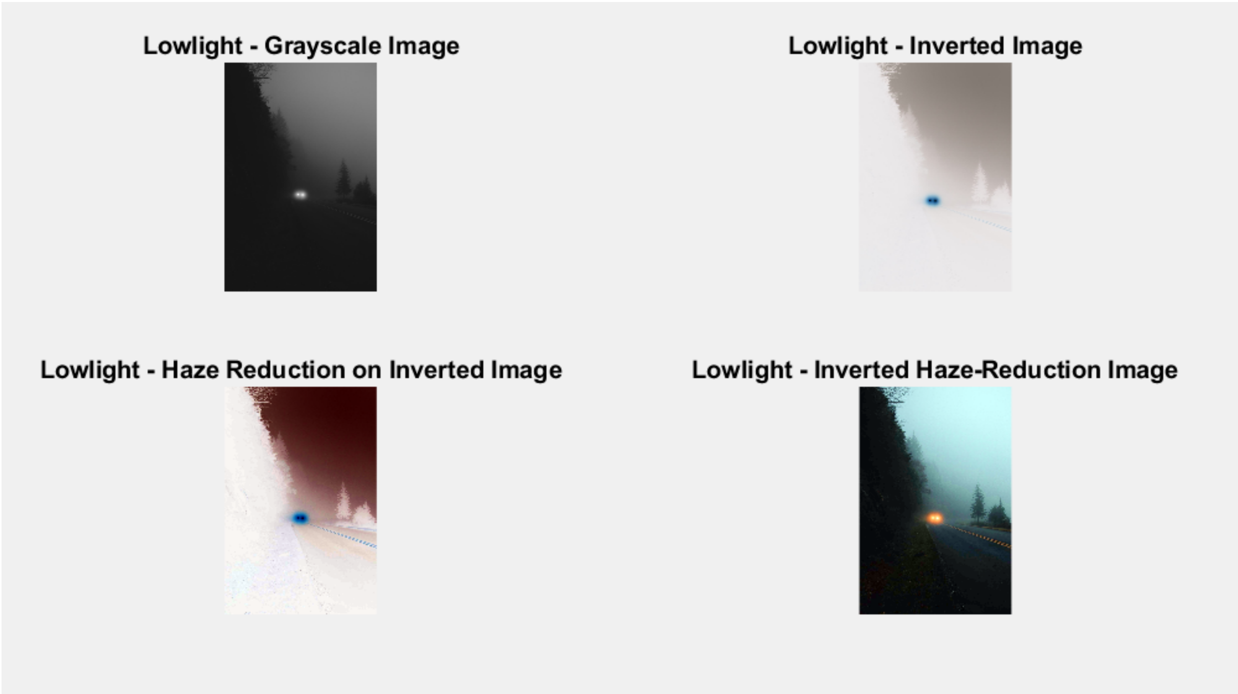

Figure: Side-by-side comparison of Lowlight image in Grayscale, inverted, haze reduction, and inverted haze reduction Images.

<u>Analysis</u>

When comparing the original image with the final inverted haze-reduced image, it is clear that the haze-reduced algorithm allows for a significant improvement in image quality, by resulting in a more contrasted and visible image.

<u>Part 3</u>

Skeletal capture and video normalization will be the project's topic.

References

Dong, X., Wang, G., Pang, Y. (Amy), Li, W., Wen, J. (Gene), Meng, W., & Lu, Y. (n.d.). FAST EFFICIENT ALGORITHM FOR ENHANCEMENT OF LOW LIGHTING VIDEO. https://www.cse.cuhk.edu.hk/~wei/papers/icme11_video.pdf

MathWorks. (n.d.). Low-Light Image Enhancement https://www.mathworks.com/help/images/low-light-image-enhancement.html

Toronto Metropolitan University. 2023. Course Content: Week 4. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. wk3 - CP8307/CPS843 - Intro to Computer Vision - F2023 (torontomu.ca)

Toronto Metropolitan University. 2023. Course Content: Week 4. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. wk4 - CP8307/CPS843 - Intro to Computer Vision - F2023 (torontomu.ca)

Toronto Metropolitan University. 2023. Course Content: Week 5. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. wk5 - CP8307/CPS843 - Intro to Computer Vision - F2023 (torontomu.ca)