RYERSON UNIVERSITY

Faculty of Engineering, Architecture and Science

Department of Electrical and Computer Engineering

Course Number	CPS 843
Course Title	Introduction to Computer Vision
Semester/Year	F2023

Instructor	Dr. Guanghui Richard Wang

ASSIGNMENT No.	2

Assignment Title	Homework 2

Submission Date	October 20th, 2023
Due Date	October 23rd, 2023

Student Name	Abdulrehman Khan
Student ID	500968727
Signature*	A.K.

**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:

Problem 1:

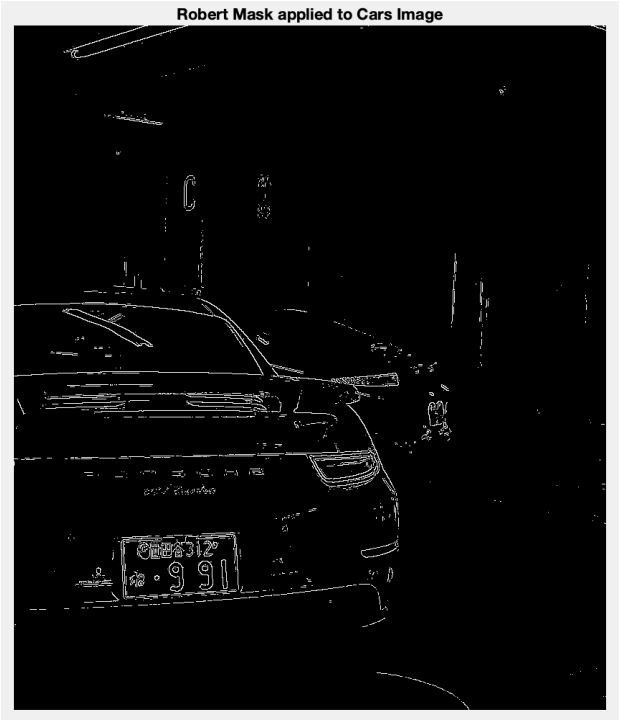
Matlab code:

```
%Problem 1

image = imread("cars.jpeg");

imageGrey = rgb2gray(image);
imshow(imageGrey)
title ('Grey-scaled Cars Image');

robertMask = edge(imageGrey, 'roberts');
imshow(robert);
title('Robert Mask applied to Cars Image');

prewittMask = edge(imageGrey, 'prewitt');
imshow(prewitt);
title('Prewitt Mask applied to Cars Image');

sobelMask = edge(imageGrey, 'sobel');
imshow(sobel);
title('Sobel Mask applied to Cars Image');
```

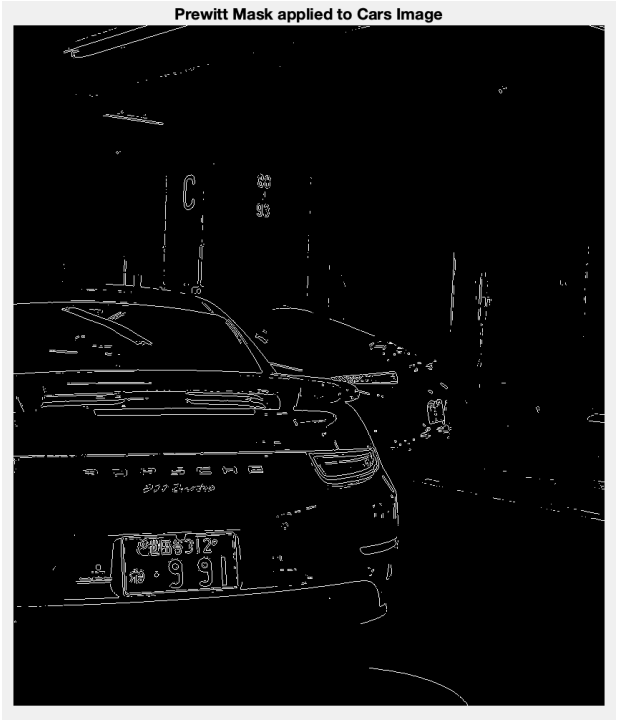
Results:

(A) Grey-scaled cars:

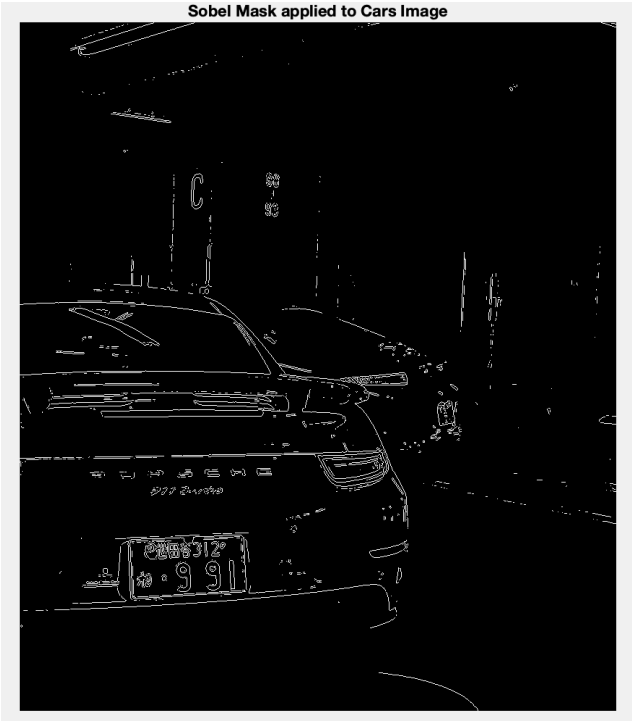
(B) Robert Mask applied to Cars Image:



(C) Prewitt Mask applied to Cars Image:



(D) Sobel Mask applied to Cars Image:



Analysis:

Robert operator: ● Simplicity: The Robert operator is the simplest among the three. ● Details: It may miss fine details and edges in the image. ● Strength: Works well for strong, well-defined edges. ● Computational Efficiency: Quick to compute, examining only four input pixels for each output pixel, using simple addition and subtraction. ● Edge Thickness: Tends to produce thicker edges in the image. ● Sensitivity to Noise: Less sensitive to noise compared to Sobel but not as effective in capturing subtle edges.	*Prewitt operator:* ● Complexity: Slightly more complex than the Robert operator. ● Edge Range: Expected to capture a broader range of edges compared to Robert. ● Balance: Offers a middle ground in terms of computational complexity and edge detection performance. ● Detects: Both vertical and horizontal edges.
Sobel operator: ● Effectiveness: Known for its effectiveness in edge detection. ● Complexity: Relatively more complex compared to Robert and Prewitt. ● General-Purpose: Often considered one of the better choices for general-purpose edge detection. ● Detects: Both vertical and horizontal edges. ● Edge Emphasis: Emphasizes pixels in the center of the mask. ● Noise Reduction: Reduces noise in the image. ● Edge Thickness: Results in thicker edges. ● Sensitivity to Noise: Less sensitive to noise compared to the Robert operator. ● Recommended Choice: Often recommended for general-purpose edge detection.	*General considerations:* ● Threshold: The effectiveness of these operators depends on the choice of parameters, including the threshold used to binarize the results. Default thresholds are used if not specified. ● Trade-off: To obtain more detailed analysis, observe the output images and evaluate the trade-off between false positives and false negatives, as well as the accuracy of edge detection for your specific image. ● Fine-Tuning: You can fine-tune the results by adjusting the threshold or using post-processing techniques like hysteresis to connect edge segments.

Problem 2

Formula and final answers:

$f(x)$	[6, 6, 6, 6, 5, 4, 3, 2, 1, 1, 1, 1, 1, 1, 6, 6, 6, 6, 6]
$f'(x) = f(x+1) - f(x)$	[0, 0, 0, -1, -1, -1, -1, -1, 0, 0, 0, 0, 0, 5, 0, 0, 0, 0, -6]
$f''(x) = f(x-1) + f(x+1) - 2f(x)$	[-6, 0, 0, -1, 0 ,0 ,0 ,0, 1, 0, 0, 0, 0, 5, -5, 0 ,0 ,0, -6]

Manual calculations:

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

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

Problem 3

Matlab Code:

```
%Problem 3

image = imread("cars.jpeg");

imageGrey = rgb2gray(image);
imshow(imageGrey)
title ('Grey-scaled Cars Image');

% Unsharp masking (default radius)
sharpen = imsharpen(imageGrey);
imshow(sharpen);
title ('Sharpened Cars Image');

% High-Boost Filter (k=1)
% No need to specify 'radius' for k=1
firstHighBoostFilter = imsharpen(imageGrey, 'amount', 1);
imshow(firstHighBoostFilter);
title('High-Boost Filter (k=1) applied to Cars Image');

% High-Boost Filter (k=5)
secondHighBoostFilter = imsharpen(imageGrey, 'radius', 5, 'amount', 5);
imshow(secondHighBoostFilter);
title('High-Boost Filter (k=5) applied to Cars Image');
```

Grey-scaled Cars Image

Sharpened Cars Image

High-Boost Filter (k=1) applied to Cars Image

High-Boost Filter (k=5) applied to Cars Image

Analysis:
Unsharp masking, demonstrated through MATLAB's imsharpen() function, involves enhancing image details by subtracting a low-pass filtered version from the original image, effectively extracting high-frequency information while reducing low-frequency components. For a value of k = 1, it subtly sharpened the car's image, highlighting edges and improving overall clarity while preserving the image information.

In contrast, high-boost filtering, with k > 1, results in a more pronounced sharpening effect. This is achieved by amplifying image edges, significantly increasing contrast, and making edges stand out prominently. The speckled appearance of the image is due to the enhanced sharpness driven by the value of k.

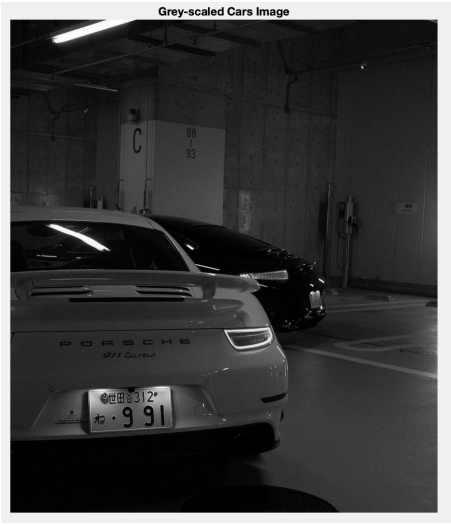
Both processes use the 'k' value to control the amplification, impacting the level of sharpening applied to our car's image. While unsharp masking offers a milder enhancement, high-boost filtering provides a more aggressive, contrast-enhancing effect.

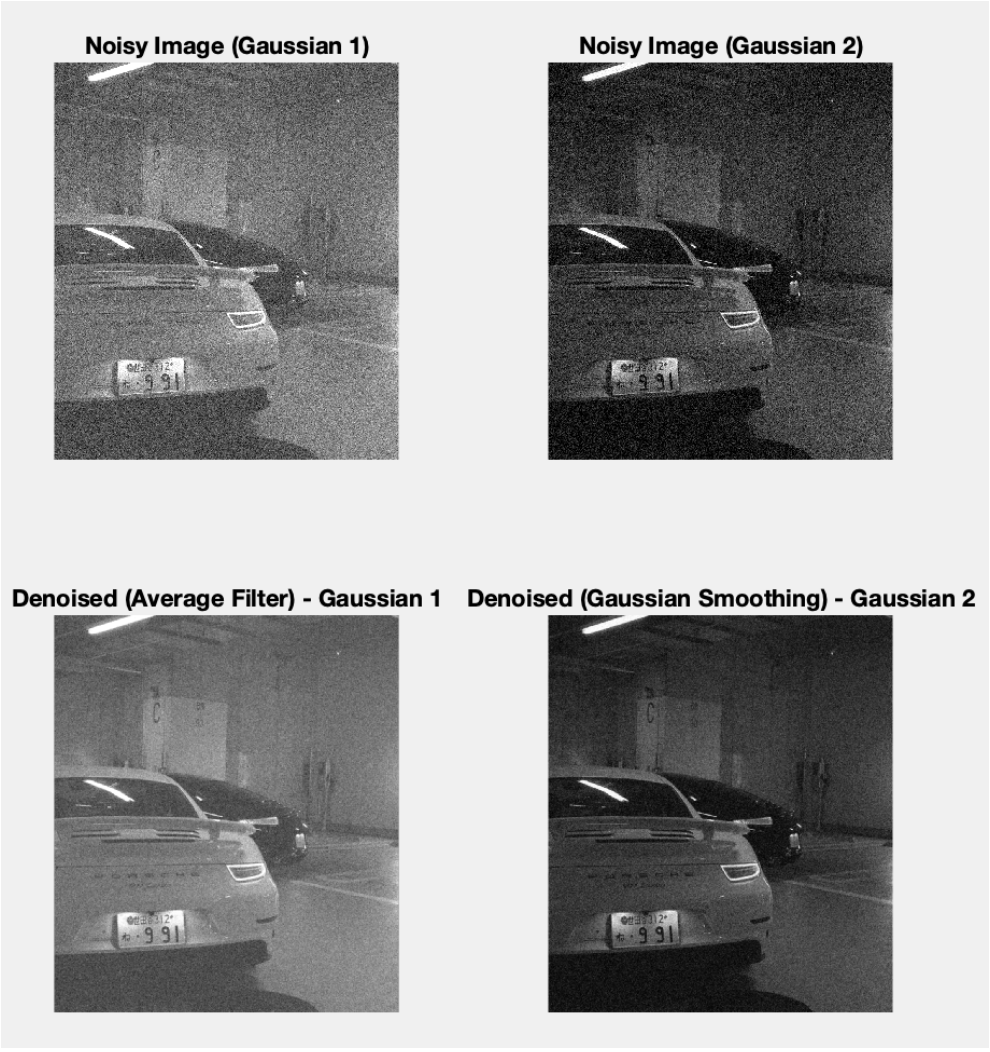
Problem 4:

Matlab code:

```
% Problem 4

image = imread("cars.jpeg");

imageGrey = rgb2gray(image);
imshow(imageGrey)
title ('Grey-scaled Cars Image');

firstGaussianLevel = 0.30;
secondGaussianLevel = 0.06;

firstGaussianImage = imnoise(imageGrey, 'gaussian', firstGaussianLevel);
secondGaussianImage = imnoise(imageGrey, 'gaussian', secondGaussianLevel);

subplot(2,2,1);
imshow(firstGaussianImage);
title ('Noisy Image (Gaussian 1)');

subplot(2,2,2);
imshow(secondGaussianImage);
title ('Noisy Image (Gaussian 2)');

gaussianImageAverage = filter2(fspecial('average',3), firstGaussianImage)/255;
subplot(2,2,3);
imshow(gaussianImageAverage);
title ('Denoised (Average Filter) - Gaussian 1');

secondGaussian = imgaussfilt(secondGaussianImage, 0.8);
subplot(2,2,4);
imshow(secondGaussian)
title ('Denoised (Gaussian Smoothing) - Gaussian 2');
```

Image Results:





Analysis:

The image titled "Noisy Image (Gaussian 1)" illustrates the application of a Gaussian filter to the initial “Grey-scaled Cars Image”, using a parameter of 0.30. As a consequence, the output is an image with a discernible amount of noise and a faded appearance. In contrast, the adjacent image labeled "Noisy Image (Gaussian 2)" with a lower value of 0.06 exhibits greater contrast and visibility, diverging from its counterpart. These manipulated images have departed from their original quality and can be enhanced by employing a specialized filter. The Average filter, when applied to the first Gaussian-processed image labeled “Denoised (Average Filter) - Gaussian 1”, calculates the pixel values by averaging the surrounding pixels, with the influence of distant pixels diminishing due to greater distance from the center pixel. Furthermore, applying the Gaussian filter to the second Gaussian-processed image labeled “Denoised (Average Filter) - Gaussian 2” results in a smoother rendition, effectively reducing both noise and low contrast introduced by the earlier filtering steps. This yields an image that more closely resembles the original grey-scaled cars image.

Question 5:

Abdulrehman Khan
500968727

1) First-order derivative with respect to x:

$$\frac{\partial f(x,y)}{\partial x} = f_x(x,y)$$

First-order derivative with respect to y:

$$\frac{\partial f(x,y)}{\partial y} = f_y(x,y)$$

Laplacian of the 2D image:

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

2) Given the 3-bit image:

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

Partial derivative with respect to x:

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

Partial derivative with respect to y:

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

3) Let's compute the Laplacian of the 3-bit image using the Laplacian mask:

$$\begin{matrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{matrix}$$

To compute Laplacian, apply mask to image via convolution:

$$\begin{matrix} -13 & 3 & -11 & 12 \\ 4 & 0 & 6 & 0 \\ -5 & -2 & -2 & 5 \\ 5 & -4 & 2 & 0 \end{matrix}$$

4) Scaling values to the range $[0, 255]$:

$$L_{\text{normalized}} = (L - \min(L)) / (\max(L) - \min(L)) \cdot 255$$

For our Laplacian image the $\min(L) = -13$ and $\max(L) = 12$

Applying the normalization formula to each pixel and rounding values to integer:

$$\begin{matrix} 0 & 255 & 38 & 231 \\ 85 & 0 & 127 & 0 \\ 21 & 63 & 63 & 191 \\ 191 & 0 & 127 & 0 \end{matrix}$$

Part 2:

The image enhancement algorithm described in the paper is designed to improve the visibility and quality of low-lighting videos. Here's a brief technical description of the algorithm:

1. **Inversion of Low-Lighting Video**: The algorithm begins by taking an input low-lighting video and inverting it. This inversion process effectively makes the dark areas appear brighter and the bright areas darker. In this operation, each pixel's color channels (R, G, B) are inverted, resulting in an inverted video.

2. **De-hazing Algorithm Application:** After inverting the low-lighting video, the algorithm applies an optimized image de-hazing algorithm. This de-hazing algorithm is based on a mathematical model that represents the relationship between the observed image, the original scene, atmospheric conditions, and the distance between objects and the camera.

3. **Temporal Correlations for Speed-Up:** To expedite the calculations, the algorithm takes advantage of temporal correlations between subsequent frames. When processing a video sequence, if the values of specific pixels in the current frame are sufficiently similar to their corresponding pixels in the previous frame, the algorithm can bypass the calculation of some parameters. This effectively reduces the computational load.

4. **Enhancement of Image:** The result of the de-hazing operation is an enhanced image with improved visibility and quality compared to the original low-lighting video.

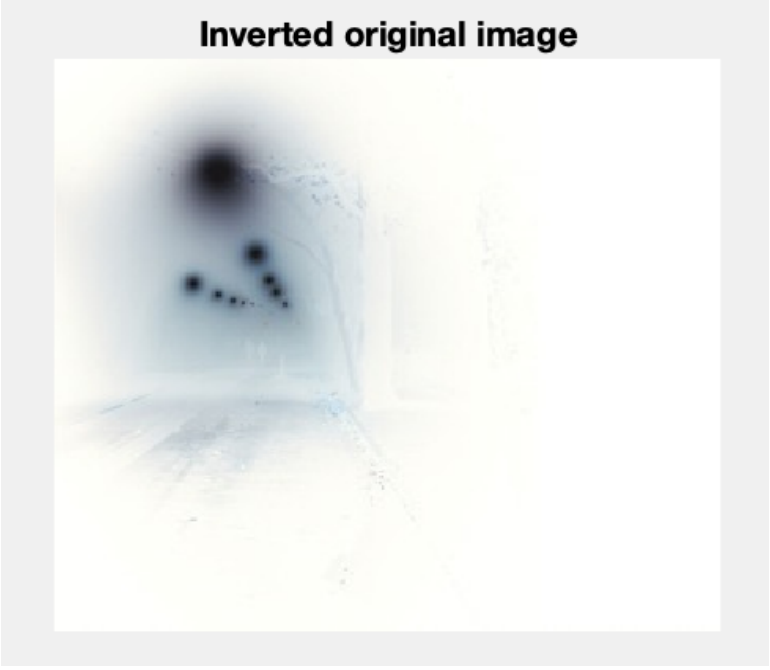
5. **Re-Inversion:** For low-lighting video, the enhanced image obtained in the previous steps is inverted again. This re-inversion operation restores the original color values of the enhanced image.

The algorithm's key innovation lies in its ability to treat low-lighting video as if it were captured in hazy conditions and then apply state-of-the-art de-hazing techniques to enhance the video. Additionally, the algorithm optimizes the calculation of de-hazing parameters and takes advantage of temporal correlations between frames, resulting in a 4x speed-up compared to traditional frame-wise enhancement approaches. The end result is an enhanced low-lighting video that is visually improved and computationally efficient.

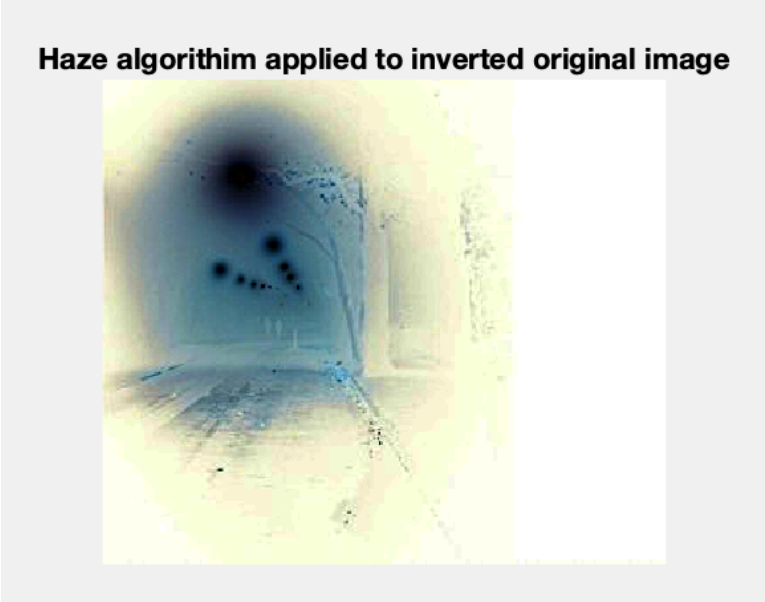
Steps:

Original image:

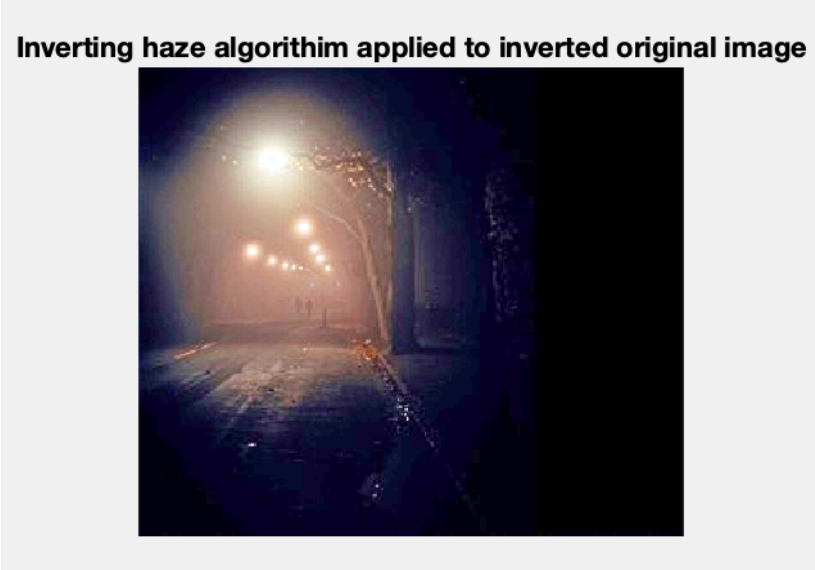
Inverted original image:



Haze algorithm applied to inverted original image:



Inverting haze algorithm applied to inverted original image:



Matlab code:

```
% Part 2

original = imread('low light.jpeg');
subplot(2,2,1);
imshow(original);
title('Original image');

originalInverted = imcomplement(original);
subplot(2,2,2);
imshow(originalInverted);
title('Inverted original image');

hazingInverted = imreducehaze(originalInverted);
subplot(2,2,3);
imshow(hazingInverted);
title('Haze algorithim applied to inverted original image');

hazing = imcomplement(hazingInverted);
subplot(2,2,4);
imshow(hazing);
title('Inverting haze algorithim applied to inverted original image');
```

Results:

- The first subplot displays the original image.
- The second subplot displays the inverted original image, which is essentially a negative of the original image.
- The third subplot displays the result of applying a haze reduction algorithm to the inverted original image. This step should enhance the image by reducing haze or fog.
- The fourth subplot displays the result of inverting the colors of the image with the haze reduction applied, essentially restoring it to a form similar to the original.

In summary, this code demonstrates a series of image processing steps involving inversion and haze reduction, with the final result being an enhanced image with reduced haze.

Part 3:

The tentative topic of our final project will be on skeleton capturing and video normalization.

References:

Dong, X., Wang, G., Pang, Y. (Amy), Li, W., Wen, J. (Gene), Meng, W., & Lu, Y. (2011). Fast Efficient Algorithm for Enhancement of Low Lighting Video. In Proceedings of the IEEE International Conference on Multimedia and Expo (ICME) (pp. 1-9). Department of Computer Science, Tsinghua University, Beijing, China. URL: https://www.cse.cuhk.edu.hk/~wei/papers/icme11_video.pdf

MathWorks. (n.d.). Low-Light Image Enhancement. MathWorks. 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. https://courses.torontomu.ca/d2l/le/content/797209/viewContent/5367444/View

Toronto Metropolitan University. 2023. Course Content: Week 5. In Computer Vision, CPS 843. Toronto Metropolitan University's Learning Management System. https://courses.torontomu.ca/d2l/le/content/797209/viewContent/5377654/View