Image Enhancement Tasks in OpenCV

Frank Cally Tabuco

Categories and Subject Descriptors

•**Computing methodologies → Image representations;**

Keywords

image enhancement, spatial domains, fourier transforms

1. INTRODUCTION

Computer vision follows the same procedures instantenously processed by humans when perceiving a certain scene or image. It involves the acquisition of an image through an eye or lens and interpretation of pixels, shapes, and colours through the brain. The main goal of computer vision is to analyze multiple images and videos to achieve human-level perception or even greater. This task would help in various fields requiring high level of image analysis such as biomedicine[5], physiology[1], and bioacoustics[4]. For computer vision, image enhancement is one crucial step involved in improving the quality and interpretability of images to be used as inputs for models or as output of results [6]. There is a wide range of image enhancement techniques developed, and each is appropriate for specific problems. In general, image enhancement can be divided into two broad fields namely: Spatial Domain Methods, and Frequency Domain Methods. Spatial domain methods employs pixel-wise operations wherein pixel values are manipulated to obtain desired image attributes. On the other hand, frequency domain methods requires conversion of images into frequency domain using the Fourier transform. All operations are then implemented on the Fourier transformed image and the resulting enhancements are then converted to an image using the Inverse Fourier transform [6][7].

In this programming assignment, the task is to use the OpenCV application to build computer programs which will employ several spatial domain enhancement techniques in various images and to demonstrate transformation properties involved in frequency domain methods. Additionally, several programs were created for familiarization of syntax and libraries within OpenCV. The rest of the paper is structured as follows: section 2 discusses the various spatial domain methods used in this paper. This section also shows the two important properties of 2D Fourier transform. In section 3, solutions to OpenCV problems are presented. Here also, we discuss the resultant images from image enhancement in comparison with original images. The last sections discuss the conclusions from these experiments and further improvements for succeeding assignments.

2. METHODOLOGY

2.1 Spatial Domain Methods

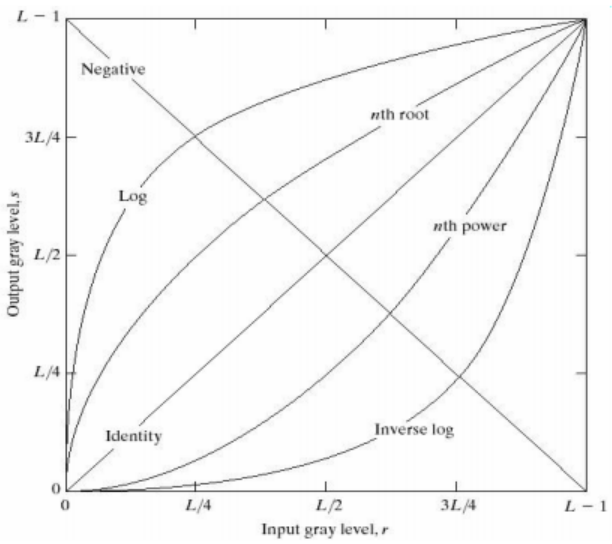

Figure 1: Basic gray level transformations.

2.1.1 *Negative transformation*

This is the most basic transformation wherein an image is produced that is equivalent to a photographic negative. This transformation enhances white or gray pixel values in dark regions of an image [7]. It can be achieved by subtracting pixel values of an image, *I(r,c)* where r represents the row and c represents the column, from the max pixel value of 255. A sample transformation of a dental image is shown

ACM ISBN XXXXXXXX.
DOI: XXXXXXX

in Figure 2. As seen from the image, pixels in dark regions were whitened while pixels in white regions are darkened.

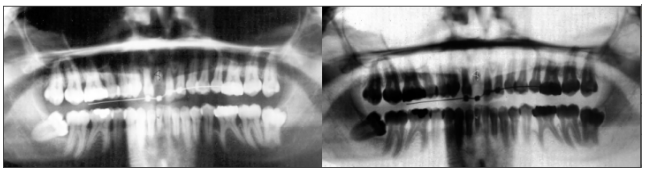

Figure 2: Negative transformation. (Original image on left, transformed on right)

2.1.2 Log transformation

Log transformations maps narrow and low-intensity pixel values to a wider and higher-intensity values [6]. The standard transformation function is denoted by $s = c * log(1+r)$, where c is a constant and r is the pixel values of an image. The scaling constant, c, is used to ensure that pixel values of an image will not exceed the maximum of 255. A sample transformation is shown in Figure 3. It can be seen that there are certain regions in the log-transformed image which shows black pixels. This is because dental images consists of pixel values of 0 (black) and 255 (white). Adding a value of 1 for each pixel should have prevented values of 0. However, for regions where pixel values are at 255, adding a value of 1 will yield a 0 pixel value.

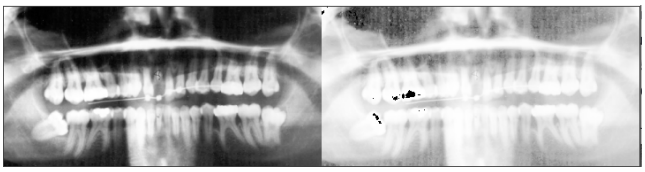

Figure 3: Log transformation. (Original image on left, transformed on right)

2.1.3 Gamma transformation

Power-law or gamma transformation is mainly used to prevent bleaching or darkening of images. Lower values of gamma results to a whitened image. As the gamma value increases, intensity values of an image decreases[7]. Figure 4 shows a sample gamma transformation. In this implementation, since the dental image has high intensity values, the gamma value is increased to a value of 2.2.

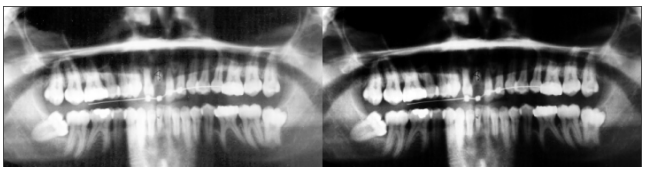

Figure 4: Gamma transformation. (Original image on left, transformed on right)

2.1.4 Piecewise transformation

Some transformations are not linear in nature. One example is piecewise transformation which is commonly used for contrast stretching[7]. Given a user-defined range of values, pixels ciorresponding to a certain range are contrasted based on minimum and maximum range values (see Figure 5).

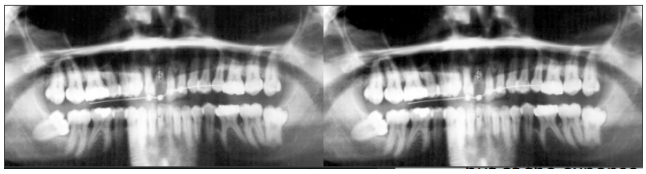

Figure 5: Piecewise transformation. (Original image on left, transformed on right)

2.1.5 Gray-level slicing

This transformation highlights gray levels of an image based on a specified range. The main idea is assigning low intensity values for regions not in the desired range, and displaying high intensity values for those within the range. This is used for emphasizing certain regions in an image[6][7]. This type of transformation is not much useful for dental images.

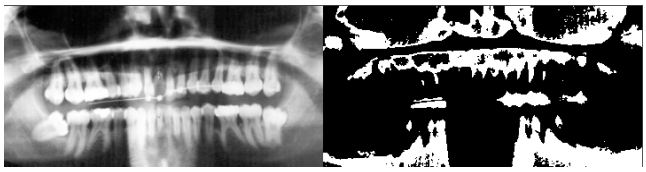

Figure 6: Gray level transformation. (Original image on left, transformed on right)

2.1.6 Bit-plane slicing

Bit-plane slicing involves modifying pixel values into their equivalent binary representation. Bits from 5 to 8 planes represent visually significant data while the other half represents finer details of an image. For this transformation, planes 1 to 4 are removed from an image. This is done by splitting the bit representations then adding only the significant planes[7].

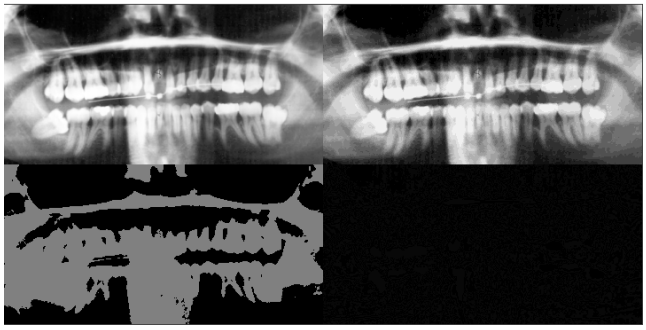

Figure 7: Bit-slicing transformation. (Original image on top-left, transformed on top-right, bit plane 7 on bottom-left, bit plane 4 on bottom-right)

2.1.7 Histogram equalization

This tranformation is a common image enhancement technique. This involves stretching intensity values of dark images to grayer levels thus equalizing the histogram of an image. The same procedure of stretching intensities is applied to whiter images. The resulting image will be clearer and more balanced than the original.

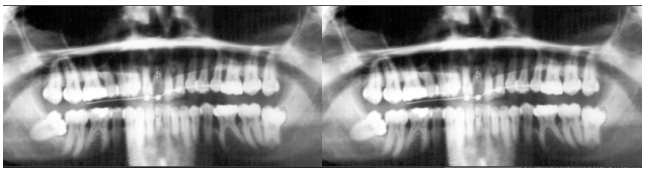

Figure 8: Histogram equalization. (Original image on left, transformed on right)

2.1.8 Blurring

Blurring is useful for images with a lot of noise. An image is blended to match the details of the background thereby reducing salt (white) and pepper (black) pixels. Salt and pepper noise are sparse pixels of white and black that distorts and image. There are several blurring techniques which can be used to handle such noises some of which are Median Blur and Gaussian Blur. Blurring is done by averaging neighboring pixels based on a kernel size[7]. This transformation is used along with other transformation methods.

2.1.9 Unsharp masking

Unsharp masking is a sharpening technique which employs blurring an image then creating a mask from the difference of the original image and blurred image. The mask is then added to the original image to create a sharper image[7]. A sample transformation is shown in Figure 9. As seen in the image below, the image on the right is sharper and less blurry compared to the original. This can be seen from the details shown on the teeth roots.

Figure 9: Unsharp Masking. (Original image on left, transformed on right)

2.1.10 Laplacian

Laplacian is a technique which employs image gradients. It utilizes filters to extract gray level discontinuities of an image. In this implementation, the Laplacian of an image is subtracted from the original image to create a sharper image[7]. As seen in Figure 10, this results to an image which is sharper than the original, but with some grains.

2.2 Frequency Domain Method

In this paper, we will be exploring the translation and rotation properties of the 2D Fourier transform. The translation property is given by:

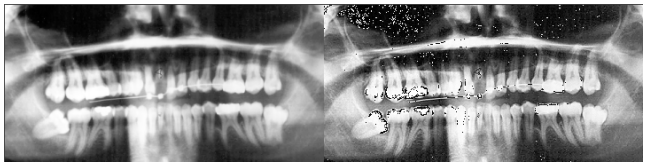

Figure 10: Laplacian. (Original image on left, transformed on right)

$$F[f(x-x_0, y-y_0] = F(u,v)e^{-j2\pi(ux_0+vy)0)/M)} \quad (1)$$

$$f(x-x_0, y-y_0) = F^{-1}[F(u,v)e^{j2\pi(ux_0+vy_0)/M}] \quad (2)$$

The above equations demonstrate that the magnitude of an image is not change when an image is moved from the center, but will undergo phase shifts. Additionally, the rotation property of the 2D Fourier transform better described in terms of polar coordinates r = x cos(θ), u = w cos(α), c = y sin(θ), and v = w sin(α). From these polar coordinates, the rotation property is given by:

$$f(x, \theta+\theta_0) = F^{-1}[F(w, \alpha+\theta_0)] \quad (3)$$

$$F[f(x, \theta+\theta_0)] = F(w, \alpha+\theta_0) \quad (4)$$

As seen from the equations, adding an angle θ from the image also rotates the corresponding frequency domain representation of the image[2].

3. RESULTS AND DISCUSSION

3.1 Additional Problems

3.1.1 Select a negative floating-point number

The OpenCV programs created used python as a programming language. Given an arbitrary floating-point number say, -4.5, the absolute value is given by 4.5, the rounded value is -4, the floor is -5, and the ceiling is -4.

3.1.2 Generate a matrix

Generating a 3x4 matrix with random integer numbers can be done using the numpy library[3]. Call the function numpy.random.randint(max value,size=(3,4)).

3.1.3 Matrix operations

Given the matrices and values shown in Figure 11, we are tasked to do some matrix operations.

1. The solution to cA + (1-c)B + d is achieved by element-wise multiplication of the scalar value c to matrix A and scalar value (1-c) to B. We then sum the results to achieve the solution shown in Table 1.

Table 1: Matrix operation solution.

(7.45, 0.85, 4.65)	(4.05, 4.75, 5.55)	(7.45, 5.45, 2.15)
(3.95, 5.35, 4.25)	(5.25, 4.25, 8.65)	(4.75, 7.45, 4.65)
(4.75, 4.65, 8.65)	(4.05, 3.55, 7.85)	(7.35, 3.25, 2.95)

2. A^{-1} using SVD. Given a matrix A, its single-value decomposition (SVD) is given by $U\Sigma V^T$, where U is a mxn orthogonal matrix, V is a nxn orthogonal matrix, and D

Table 2: Inverse of a Matrix using SVD.

(0.11417323, -0.05511811, 0.04724409)	(-0.16929134, 0.04724409, 0.1023622)	(0.15748031, 0.16535433, -0.14173228)
(-0.26545455, 0.08727273, 0.13818182)	(0.16, -0.08, 0.04)	(0.07636364, 0.09818182, -0.09454545)
(-0.13636364, 0.13636364, 0.09090909)	(0.46590909, -0.71590909, 0.27272727)	(-0.10227273, 0.35227273, -0.18181818)

$$A = \begin{pmatrix} (6,0,2) & (2,6,5) & (9,7,1) \\ (1,6,4) & (5,4,9) & (6,9,2) \\ (3,5,9) & (5,3,7) & (8,3,3) \end{pmatrix}$$

$$B = \begin{pmatrix} (9,1,9) & (7,0,5) & (2,0,3) \\ (9,2,3) & (4,3,6) & (0,2,9) \\ (7,2,6) & (0,3,8) & (4,2,1) \end{pmatrix}$$

$$c = 0.7$$

$$d = 0.55$$

Figure 11: Matrices and scalar values.

is an nxn diagonal matrix consisting of the singular values. From this, the pseudoinverse A^{-1} is given by $V\Sigma^{-1}U^T$[8]. Given this formulation, the inverse of A can easily be solved (see Table 2).

3. We can get the eigenvalues of B by solving the characteristic polynomial det(B-αI) = 0, where det is the determinant. The eigenvalues are shown in Table 3.

Table 3: Eigenvalues of B.

(11.75011644, 1.10052949, -0.85064593)
(0.89725152, 8.03460809, 12.06814039)
(11., -3., 3.)

4. Solving the equation Ax = b entails getting the inverse of A, which we have already computed above. Multiplying both sides by the inverse of A results to $x = A^{-1}(b)$. For this 3D matrix, there are three possible values of x to get b. Each row in Table 3, represents possible values of x.

5. Creating a triangle given two points (30,10) and (60,40) can be achieved using the rectangle function in OpenCV. This will display a rectangle with a user-assigned color. Converting this image into grayscale requires a simple conversion function in OpenCV. The resulting images are shown in Figure 12.

3.1.4 *Reading Videos*

Reading videos in OpenCV requires assigning a variable to the VideoCapture function. An input in this function can be real-time videos through cameras or filepaths to a corresponing video file. OpenCV can then read and display each frame in the video using a *while Loop*. Trackbars are used to track the current frame displayed and also to pause/play the video. Below is a snapshot of the created video reader.

3.1.5 *Displaying images and pixel values*

Displaying an image is simpler than reading video files. However, for this program we need to alsp record mouse-pointer coordinates and display the corresponding pixel value

Figure 12: Rectangle images in OpenCV.

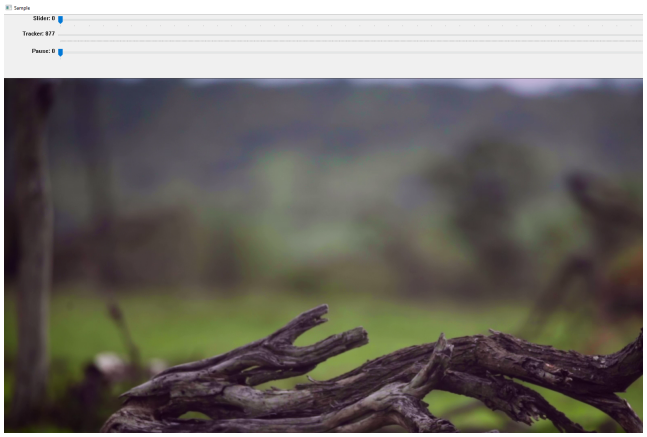

Figure 13: Sample video display.

at that particular point. This can be achieved by creating a function and using the EVENT_LBUTTONDOWN function in OpenCV. The event function will retrieve the pointer coordinates. Given this coordinate, we can get the pixel values by mapping the coordinates from the image (see Figure 14).

3.1.6 *Recording webcam feeds*

This approach is similar to the previous video reading program. The difference is the use of three different frames which displays colored, grayscale, and canny video feeds from a webcam. For this program, instead of calling a specific file in the VideoCapture() function, we input a value of 0 (or another number depending on the number of cameras). Afterwards, we stack video feeds of colored, grayscale, and canny. It should be noted here, that colored images has a size of 3, while grayscale and canny has only 2. Some manipulations and conversions are necessary to achieved the desired display (see Figure 15).

3.2 Spatial domain results

The butterfly image in Figure 16 shows results of different spatial domain methods. The original image on the top left

Table 4: Possible values of x.

(0.14566929, 0.50787402, 0.41338583)	(0.23228346, -0.6496063, 0.14566929)	(1.06299213, 0.97637795, 1.25984252)
(0.36727273, -1.01454545, 0.2)	(0.08, 0.68, 0.4)	(0.53818182, 0.48363636, 0.6)
(0.90909091, -0.31818182, 0.95454545)	(-2.52272727, 1.17045455, -1.76136364)	(1.43181818, 0.01136364, 1.21590909)

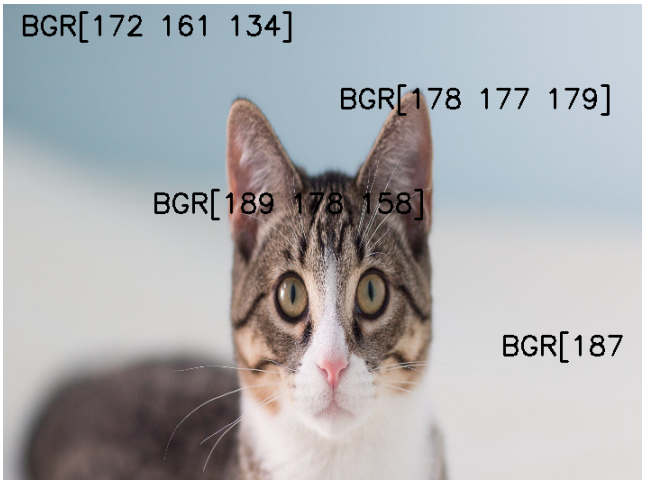

Figure 14: Pixel values at certain coordinates.

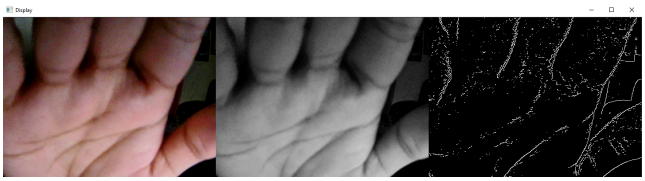

Figure 15: Colored, grayscale, and canny displays.

shows that the gray-level intensities of the main subjects almost have the same values as the back ground. Because of this, the butterfly and the flower seems to be blending with the background. Using several transformations it seems that histogram equalization produced the best image enchancement quality. Initially, we did not notice that the butterfly is sitting on top of a flower. Only after enhancing the image that the details of the butterfly and flower stood out. Another good enhancement is seen on the third row, second column. This image used median blur first then equalized the histograms. Using blurring helped in removing the salt noise and equalization improved the distinct intensities of the image.

Figure 17 shows image enhancements on a cell image. The original image seems to be a bit blurry with some edges of cells being difficult to distinguish. With no knowledge about cells, the negative transformation seemed to provide the enhancement needed. It separated regions within cells and outside the cells by highlighting cell edges. However, after consultation with some colleagues in the medical field, the unsharp masking technique yielded the best results. Not only did it improve the edges, it also showed better cellular details compared to the original.

Most of the transformations shown on Figure 18 have already been shown in the methodology section. The original dental image shows a blurry quality in certain teeth and

Figure 16: Transformations of butterfly image. For each row from left to right: original image, negative transform, logarithmic transform, gamma transform, piecewise transform, gray-scale transform, bit-plane slicing, histogram equalization, bit-plane slicing & histogram equalization, median Blur then histogram equalization, unsharp masking, subtraction of Laplacian image.

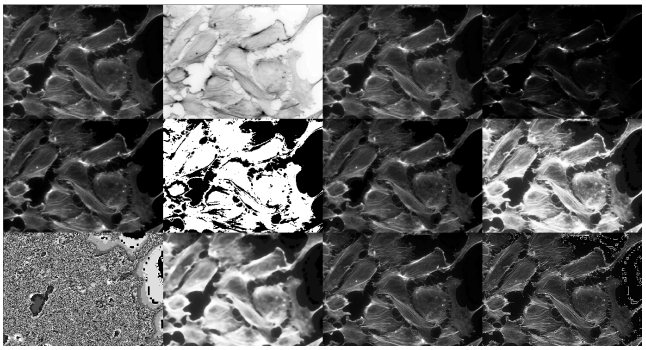

Figure 17: Transformations of cells image. For each row from left to right: original image, negative transform, logarithmic transform, gamma transform, piecewise transform, gray-scale transform, bit-plane slicing, histogram equalization, bit-plane slicing & histogram equalization, median Blur then histogram equalization, unsharp masking, subtraction of Laplacian image.

roots region. At first glance, the transformations for this image yielded similar results as can be seen with piecewise, bit-plane slicing, and histogram equalization images. Since the original image is bleached, darkening the image could yield better results. The gamma transofmation on the top-right corner improved the image quality by clearing teeth and canal regions of unneccessary white pixels. The image now shows better details of teeth and roots regions especially

on the center part of the original image.

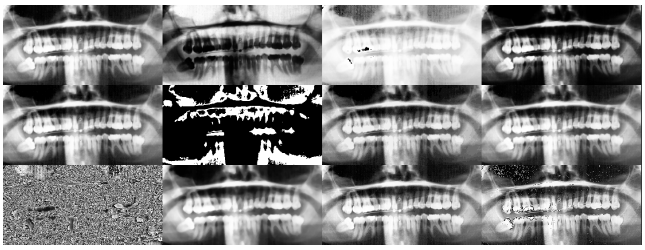

Figure 18: Transformations of dental image. For each row from left to right: original image, negative transform, logarithmic transform, gamma transform, piecewise transform, gray-scale transform, bit-plane slicing, histogram equalization, bit-plane slicing & histogram equalization, median Blur then histogram equalization, unsharp masking, subtraction of Laplacian image.

The image showing a mom and her kids has a lot of salt and pepper noise. Previously mentioned techniques did not remove any noise from the original picture. For this image, blurring techniques are necessary to enhance the image quality. Gaussian blur is used to normalize the intensities of an image. As shown third row and third column, it completely removed the salt noise while still maintaining the edge features i.e. human faces and shirt prints. On the other hand, median blur replaces pixels by the median value of its neighbors and is used to deal with both salt and pepper noise. The image on the bottom right, completely removed the salt and pepper noise from the original image. However, edges of the image is lower compared to Gaussian blur but only by a small fraction.

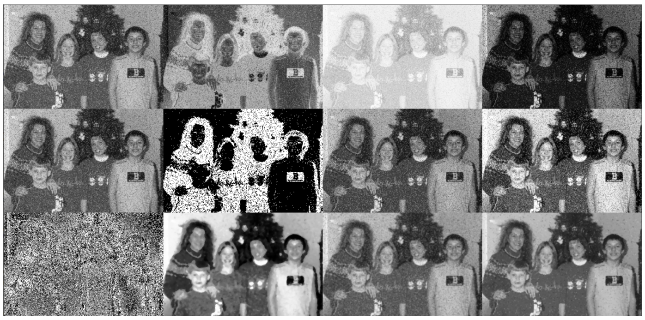

Figure 19: Transformations of mom and kids image. For each row from left to right: original image, negative transform, logarithmic transform, gamma transform, piecewise transform, gray-scale transform, bit-plane slicing, histogram equalization, bit-plane slicing & histogram equalization, median Blur then histogram equalization, Gaussian blur, median blur.

Overall, the transformed building images on Figure 20 showed sharper image quality than the original. The main difference between the original and the transformed images are the intensities in the leaves of the tree. The original canopy has a monotonous color but with unsharp masking there seems to be regions which are brighter than the rest. These regions previously not seen on the original image are inherent to the picture and not just a product of the transformation (the light patches at the base of the tree are proof of difference in lighting at the canopy). The transformed image using Gaussian blur clearly distinguished light and shadow regions of the canopy which cannot be seen clearly from the original image.

Figure 20: Unsharp masking using blur and Gaussian blur. From left: original image, using blur function, and using Gaussian blur.

3.3 2D Fourier transform properties

The translation property of the 2D Fourier transform states that the magnitude of an image willl not change when an image's position is change. Only a phase shift will occur based on the formulation described in equations 1 and 2. On the other hand, the rotation property describes the angular change of an image when the Fourier transformed image is rotated by an angle of θ, and vice versa. Looking at Figure 21, the original image magnitude and phase shift are shown in the first column. The magnitude of the original image is the same as the magnitude of the moved pictures in columns 2 and 3. However, the phase spectrum is different for each. While the original image has a vertical line at the center of the image, this is not seen on the phase spectrums of the moved images. The phase spectrums are fuzzy in nature. This coincides with the tranlation property that only the magnitude remains the same but the image still undergoes a phase shift. In the last column, the image is rotated by an angle of 90. Rotating the original image also rotated the magnitude and phase spectrums clearly seen on the moved vertical line from the original image - now it is horizontal! This proves that the rotation property holds true for 2D Fourier transformed images.

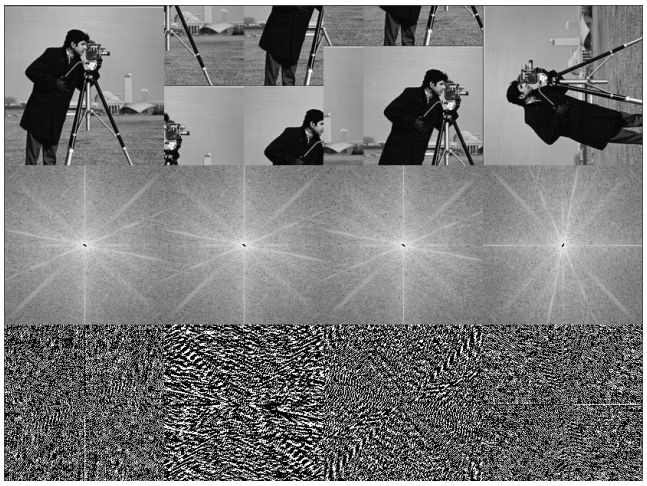

Figure 21: Magnitude and phase spectrums of images. From top row to bottom: images, magnitude spectrum, phase spectrum

4. CONLUSION

In this paper, we have seen the different image enhancement methods in the spatial domain. We have shown that these methods have specific use and some are better in improving the interpretability of an image. Combinations of image enhancement methods such as unsharp masking, bitplane slicing and histogram equalization, and subtraction of Laplacian gradients are possible and would lead to better results. However, the criteria for evaluating better images in this paper is subjective. Each methods has its own strength and use which is entirely dependend on the goal of a research. No one method is better than the other in showing image details. Some methods, like gray-level slicing, did not produce better images but are useful in segmenting or highlighting certain regions of an image. It is up to the researcher on how he/she will utilize these techniquese for his/her own benefit.

5. FURTHER WORKS

One major improvement should be focused on optimizing the button functionality in OpenCV. There have been several attempts in compiling the library with QT support to enable the button functions. However, no attempt yielded the desired outcome. Thus, a trackbar is utilized instead but has a major downside. It pauses on a specific frame, but when clicked to play it need an additional key press to continue. Better understanding of the OpenCV library is needed in future works.

6. REFERENCES

[1] Cho, Y. Rethinking eye-blink: Assessing task difficulty through physiological representation of spontaneous blinking. *In Proceedings of CHI Conference on Human Factors in Computing Systems (CHI'21)* (May 2021).

[2] Gonzalez, R. C., and Woods, R. E. *Digital Image Processing (3rd Edition)*. Prentice-Hall, Inc., USA, 2006.

[3] Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'io, J. F., Wiebe, M., Peterson, P., G'erard-Marchant, P., Sheppard, K., Reddy, T., Weckesser, W., Abbasi, H., Gohlke, C., and Oliphant, T. E. Array programming with NumPy. *Nature 585*, 7825 (Sept. 2020), 357–362.

[4] Lasseck, M. Acoustic bird detection with deep convolutional neural networks. In *Detection and Classification of Acoustic Scenes and Events 2018* (2018).

[5] Lopez, F., Varela, A., Hinojosa, O., Mendez, M., Trinh, D.-H., ElBeze, J., Hubert, J., Estrade, V., Gonzalez, M., Ochoa, G., and Daul, C. Assessing deep learning methods for the identification of kidney stones in endoscopic images, 2021.

[6] Maini, R., and Aggarwal, H. A comprehensive review of image enhancement techniques, 2010.

[7] Naval, P. Image enhancement in the spatial domain, 2021.

[8] Strang, G. *Linear Algebra and Its Applications (Fourth Edition)*. 2006.