Investigating the Self-Similarity of Images

Haoyue Chen

Supervisor: Edward R. Vrscay
(Applied Mathematics)

Co-Supervisor: Kostadinka Bizheva (Physics)

University of Waterloo

PHYS 437A research project

January 23, 2020

1 Abstract

Image self-similarity plays a significant role in image processing, forming the basis for many imaging techniques, such as Fractal image coding, Non-local means denoising and non-local methods of enhancement (e.g. super-resolution).

Starting from images being represented by functions, his thesis mainly discusses about self-similarity in image processing in terms of the pixel representation of images, which can be showed by two different ways: Mean Squared Error (MSE) and the Structural Similarity (SSIM) Index. After defining these two measures, we shall use them to estimate the similarities between subblocks of an image with respect to affine greyscale transformation. We employ these particular types of affine transformation. The final result is that MSE and SSIM show different degrees of self-similarity. From a visual perspective, SSIM is shown to be the better measure.

Contents

List of Figures

2 Introduction

The motivation for my research is that the degree that pixel-blocks of an image can be well approximated by other pixel blocks of itself is the theoretical basis of a large portion of efficient non-local image processing techniques like nonlocal-means denoising, restoration, compression, super-resolution and fractal image coding. [1, 2, 3, 4]

Generally, we call this property of images self-similarity, and it can be possessed by many fractal sets, i.e. they can be expressed as unions of contracted copies of themselves. Here are some examples of self-similarity as follows:

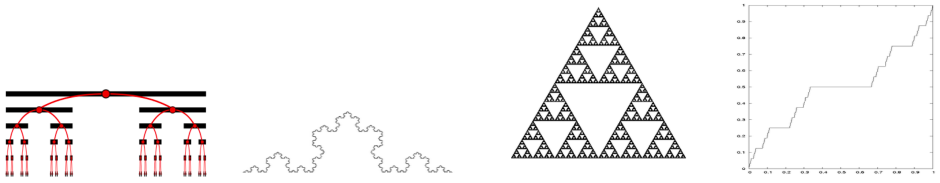


Figure 1: Some examples for self-similarity: (1) Ternary Cantor set; (2) von Koch curve; (3) Sierpinski triangle; (4) Devil's staircase function.

A similar idea of self-similarity exists in image processing, which makes it worth studying. We aim to use two different distances to measure self-similarity, then see what are the advantages and shortcuts for both methods, which is exactly my motivation for this research project.

3 Mathematical Background

This section contains some important mathematical background for this thesis. Images are represented as functions, and two kinds of measures are defined for the comparison of images.

3.1 Images as functions

Images may be considered as two-dimensional signals. Further more, digital images can be represented as matrices. Given a $m \times n$ grid I, when we approximate an $m \times n$ image, it will be represented mathematically by a function, i.e. $u(i, j)$, where (i, j) is the point of an image. The matrix entry $u(i, j)$ will denote the greyscale value at pixel (i, j) For a black-and-white image, $u(i, j)$ is usually a non-negative value, called the "greyscale value" that represent the greyness of the image at (i, j). And a colour image can be represented by a three-dimensional function, each point is defined three colour values as red, green and blue, and combination of three primary colours produces the colour at the point (i, j).

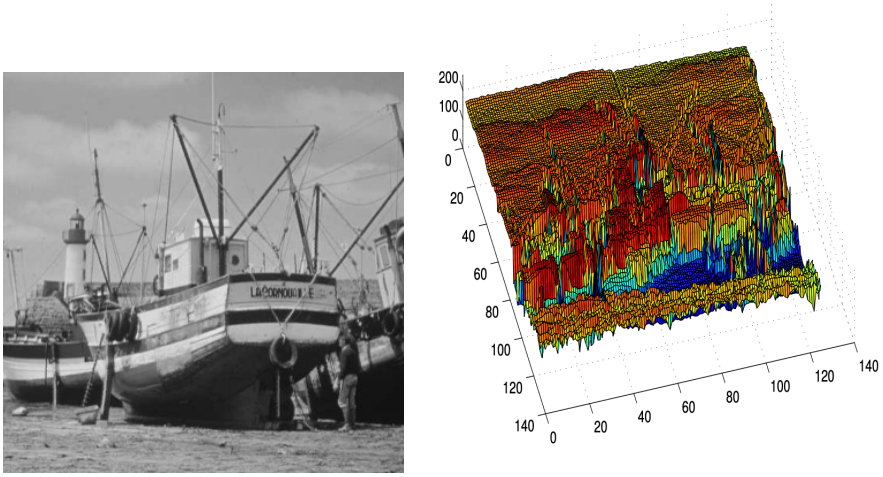


Figure 2: Greyscale and the red-blue spectrum vision of *BOAT* image

The image at the left of Figure 2 is "*BOAT*", the standard test-image, a 512×512−pixel digital image, 8 bits per pixel. Each pixel assumes one of 256 greyscale values between 0 and 255, shows the graph of the image function $u(i, j)$. The red-blue spectrum of colours is used to characterize

function values: Higher values are more red, lower values are more blue.

For an 8 bit-per-pixel image, its greyscale value $u(i,j)$ should between 0-255. In our computations, however, we normalize the value of $u(i,j)$ to be between 0 and 1: the value 0 represents black while the value 1 represents white. In this thesis we mainly consider greyscale images.

3.2 L^2 (Euclidean) Distance

It is important to quantify distances between images in order to compare them and to compute approximation errors. We used two distance measures in this thesis. In what follows, L^2 distance will be introduced first.

Let $x = (x_1, x_2, ..., x_N)$ and $y = (y_1, y_2, ..., y_N)$, then the L^2 distance between x and y can be defined as

$$d(x,y) = \left(\sum_{k=1}^{N} (x_k - y_k)^2 \right)^{\frac{1}{2}}. \tag{1}$$

One of the most widely used variants of the L^2 distance in imaging processing is the root mean error (RMSE) defined below,

$$RMSE(x,y) = \left(\frac{1}{N} \sum_{k=1}^{N} (x_k - y_k)^2 \right)^{\frac{1}{2}}. \tag{2}$$

Also the mean squared error (MSE) is

$$MSE(x,y) = RMSE(x,y)^2 = \frac{1}{N} \sum_{k=1}^{N} (x_k - y_k)^2. \tag{3}$$

3.3 The Structural Similarity (SSIM) Index

The Structural Similarity Index, so called SSIM index is another useful index for calculating error. We can set a definition for SSIM function, for $x, y \in \mathbb{R}$ is defined as follows [5],

$$S(x,y) = \frac{2\bar{x}\bar{y} + C_1}{\bar{x}^2 + \bar{y}^2 + C_1} \frac{2s_x s_y + C_2}{s_x^2 + s_y + C_2} \frac{s_{xy} + C_3}{s_x s_y + C_3}. \tag{4}$$

where the parameters C_1, C_2, C_3 are small positive constants.

3.4 Introduction to the idea of self-similarity of images

We have the definition of self-similarity that regions (subblocks) of an image are similar (in greyscale or colour values) to other regions of the image.

Self-similarity is very important in image processing for the reason that any imaging techniques use self-similarity as the basis to form an integral part, for example, fractal image coding, non-local means denoising and non-local methods of enhancement like super-resolution. [1, 2, 3, 4]

In the next section, two methods to calculate self-similarity will be introduced.

4 Self-similarity of images with respect to the L^2 (Euclidean) distance

4.1 Introduction

Image self-similarity will be characterized in terms of L^2 distances between image subblocks. Let u and v represent two $n \times n$ subblocks of an image, where subblock u stands for the range block and subblock v stands for the domain block. Then we can examine the self-similarity in term of L^2 distance by compiling the errors in approximating each subblock u with all other subblocks v with the form,

$$u \approx \alpha v + \beta. \tag{5}$$

We consider best approximation of subblock v by affine transformation of subblock u, which can be written as

$$v_i \approx \alpha u_i + \beta \ , \qquad 1 \leq i \leq N, \tag{6}$$

where u_i and v_i denote the greyscale values at each pixel.

The squared error is defined as

$$\Delta(\alpha,\beta)=\sum_{i=1}^{N}(u_i-\alpha v_i-\beta)\ , \qquad 1\le i\le N \tag{7}$$

Here we consider 3 cases as follows:

1. Case 1: $\alpha=1$, $\beta=0$; No greyscale transformations.

2. Case 2: $\alpha=1$, $\beta=\bar{u}-\bar{v}$; Greyscale shift only.

3. Case 3: $\alpha=\frac{s_{uv}}{s_v^2}$, $\beta=\bar{u}-\alpha\bar{v}$; Scaling plus greyscale shift.

Here,

$$\bar{u}=\frac{1}{N}\sum_{i=1}^{N}u_i,$$
$$s_{uv}=\frac{1}{N-}\sum_{i=1}^{N}(u_i-\bar{u})(v_i-\bar{v}),$$
$$s_u^2=\frac{1}{N-1}\sum_{i=1}^{N}(u_i-\bar{u})^2.$$

In our numerical computations, we shall use non-overlapping 8×8 pixel blocks for convenience. The root mean squared error (RMSE) is applied to report the distance between subblocks. For representations of self-similarity in figure, the standard 8-bit 512×512 *LENA* and *PEPPERS* image are used.

Figure 3: *LENA* (left) and *PEPPERS* (right)

4.2 Case 1 and Case 2

In Case 1, we take $u \approx v,\ u_i \approx v_i,\ 1 \leq i \leq N$, then the error becomes

$$\begin{aligned} \Delta_1 &= \sum_{i=1}^{N} (u_i - v_i)^2 \\ &= (N-1)[s_u^2 + s_v^2 - 2s_{uv} + N(\bar{u} - \bar{v})^2]. \end{aligned} \tag{8}$$

Applying the equation in to the code (APPENDIX A) in MATLAB, the following figures of Case 1 are obtained,

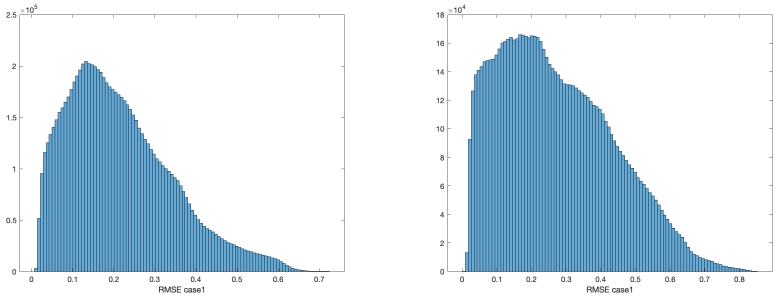


Figure 4: Case 1 error for L^2 distance of *LENA* (left) and *PEPPERS* (right).

For the Case 2, the approximation is $u \approx v + \beta,\ u_i \approx v_i + \beta,\ 1 \leq i \leq N$, then this best approximation is achieved when $\beta = \bar{u} - \bar{v}$, as we derived before

$$\begin{aligned} \Delta_2 &= \sum_{i=1}^{N}(u_i - v_i - \beta)^2, \\ \frac{\partial \Delta_2}{\partial \beta} &= -2\sum_{i=1}^{N}(u_i - v_i - \beta). \end{aligned} \tag{9}$$

Therefore

$$\begin{aligned} \sum_{i=1}^{N}(u_i - v_i - \beta) &= 0, \\ \sum_{i=1}^{N}(u_i - v_i) &= N\beta, \end{aligned} \tag{10}$$

and

$$\beta = \frac{1}{N}\sum_{i=1}^{N}(u_i - v_i) = \bar{u} - \bar{v}. \tag{11}$$

The error of Case 2 is

$$\begin{aligned} \Delta_2 &= \sum_{i=1}^{N}(u_i - v_i - \bar{u} + \bar{v})^2 \\ &= (N-1)[s_u^2 + s_v^2 - 2s_{uv}]. \end{aligned} \tag{12}$$

Once again, we use the code to produce the results in the figures below

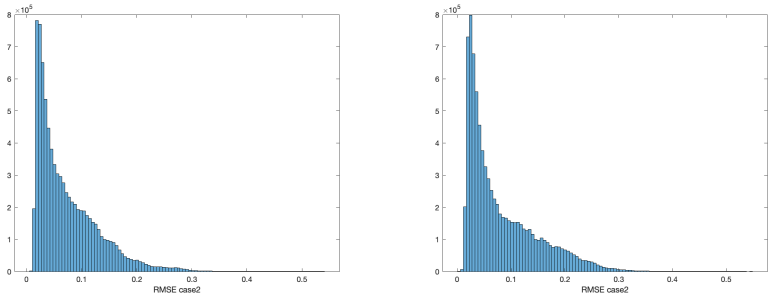


Figure 5: Case 2 error for L^2 distance of *LENA* (left) and *PEPPERS* (right).

From the comparison of the figures of Case 1 and Case 2, it is clear that there are big improvements between them. The histogram of Case1 is closer to zero implying many more smaller errors than Case 1. This is to be expected since there is one more parameter added in Case 2, implying that

$$0 \leq \Delta_2 \leq \Delta_1. \tag{13}$$

4.3 Case 3

In the previous section, we have discussed the Case 1 and Case 2. In this section we introduce the Case 3 in L^2 distance. Once again, we start with the definition of Case3, the approximation we assume

$$u \approx \alpha v + \beta,$$

so that

$$u_i \approx \alpha v_i + \beta,\ 1 \leq i \leq N.$$

To work out the values of α and β, from the best approximation, we need to minimize the squared error defined before,

$$\Delta = \sum_{i=1}^{N} (u_i - \alpha v_i - \beta)^2 \tag{14}$$

then impose the stationary conditions,

$$\begin{aligned} \frac{\partial \Delta}{\partial \alpha} &= 0, \\ \frac{\partial \Delta}{\partial \beta} &= 0. \end{aligned} \tag{15}$$

which yields the following system of equations in α and β

$$\begin{aligned} \left(\sum_i v_i^2 \right) \alpha + \left(\sum_i v_i^2 \right) \beta &= \sum_i u_i v_i, \\ \left(\sum_i v_i \right) \alpha + n\beta &= \sum_i u_i. \end{aligned} \tag{16}$$

The solution of the system is

$$\begin{aligned} a_{L^2} &= \frac{n\sum_i u_i v_i - (\sum_i u_i)(\sum_i v_i)}{n\sum_i v_i^2 - (\sum_i v_i)^2} = \frac{s_{uv}}{s_v^2}, \\ b_{L^2} &= \frac{(\sum_i v_i^2)(\sum_i u_i) - (\sum_i u_i v_i)(\sum_i v_i)}{n\sum_i v_i^2 - (\sum_i v_i)^2} = \bar{u} - \alpha\bar{v}. \end{aligned} \tag{17}$$

In summary, the optimal values of α and β from the approximation are

$$\begin{aligned} a_{L^2} &= \frac{s_{uv}}{s_v^2}, \\ b_{L^2} &= \bar{u} - \alpha\bar{v}. \end{aligned} \tag{18}$$

Then the error is given by

$$\Delta_3 = (N-1)[s_u^2 - \frac{s_{uv}^2}{s_v^2}]. \tag{19}$$

Using these result in MATLAB code, we obtained the following histograms of Case 3 approximation errors.

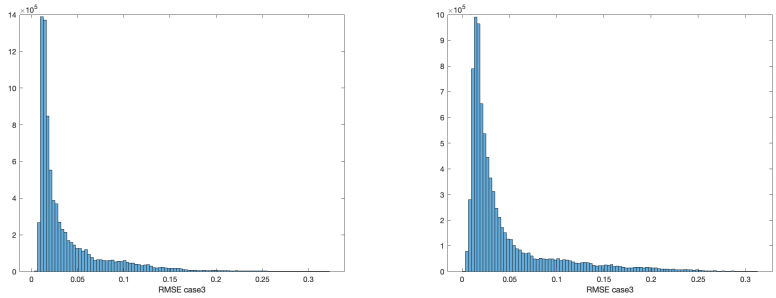


Figure 6: Case 3 error for L^2 distance of *LENA* (left) and *PEPPERS* (right).

Clearly, there is an improvement between the histograms from Case 2 to Case 3 to zero, which indicates that the approximations become better.This is expected, since the Case 3 approximation involves an optimization over two parameters, one more than Case 1.

Then the final result of errors between Case1, Case2 and Case 3 are

$$0 \leq \Delta_3 \leq \Delta_2 \leq \Delta_1. \tag{20}$$

In the next figures, we show the Case 1, Case 2 and Case 3 histograms in one plot for each image. The differences between the cases are clearly seen:

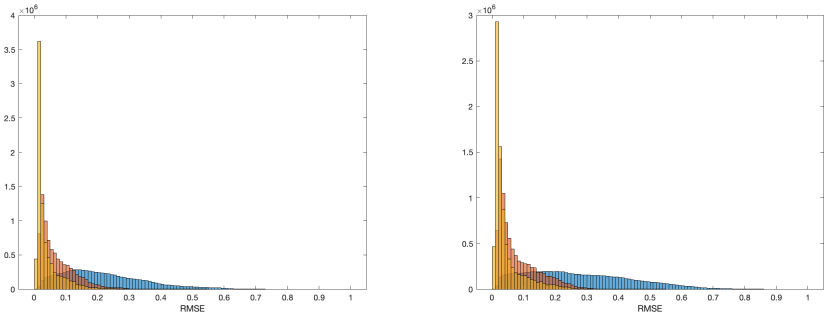


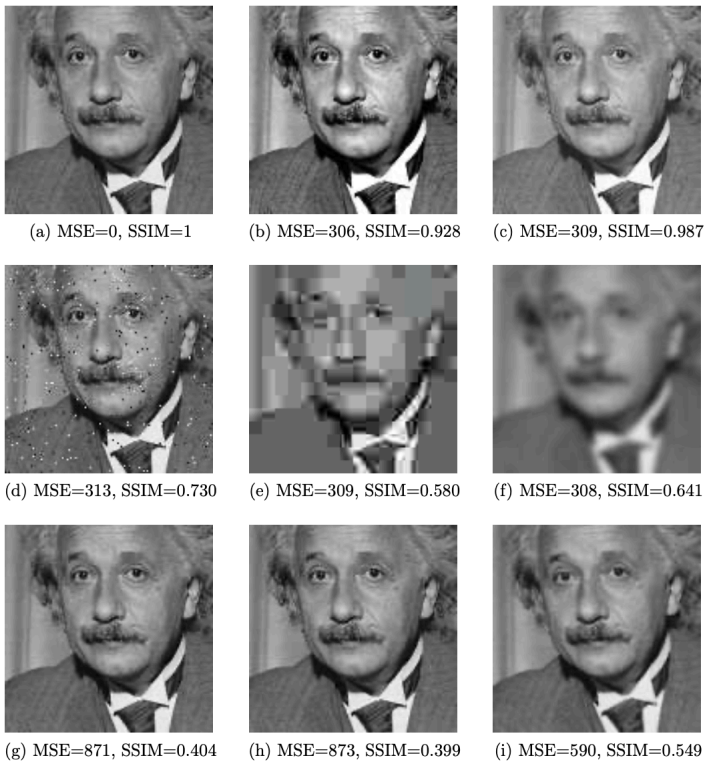
Figure 7: Case 1,2,3 error for L^2 distance of *LENA* (left) and *PEPPERS* (right).

Definitely, a higher the degree self-similarity with smaller error will bring the peak of a histogram closer to zero.

5 Self-Similarity With the Structural Similarity Index

MSE is one of the most dominant methods to calculate the self-similarity, that's why we use a whole section to introduce the method of using RMSE to measure self-similarity of image. However there is still some insurmountable weakness of using MSE to investigate self-similarity, since it is not good for image visual quality.

To explain this I will use a famous Einstein image with some variations, performed by different MSE and SSIM values. For these nine images, they respectively represent original image, mean contrast stretch, luminance shift, impulsive noise contamination, JPEG compression, blurring, spatial shift to the right and left, and the last one counter-clockwise rotation.



(a) MSE=0, SSIM=1 (b) MSE=306, SSIM=0.928 (c) MSE=309, SSIM=0.987

(d) MSE=313, SSIM=0.730 (e) MSE=309, SSIM=0.580 (f) MSE=308, SSIM=0.641

(g) MSE=871, SSIM=0.404 (h) MSE=873, SSIM=0.399 (i) MSE=590, SSIM=0.549

Figure 8: Comparison of MSE and SSIM values for an Einstein image and its variations. (Taken from [5].)

The images (b) to (f) are degraded versions of Einstein with roughly same MSE (300). Some of these images look better than others. Images (g) to (i) are degraded version of Einstein with much larger MSE, but which have better visual quality. We must conclude that MSE is not a good measure of image quality.

In the very beginning section, I introduced the structural similarity (SSIM) index, indicates another way to calculate self-similarity, and from the "Einstein" image and SSIM values, (b) to (f) images are degraded versions with roughly same MSE (300), but with much different SSIM values. That means the quality of the image is better reflected in the SSIM values. This indicates that SSIM is a better measure of image quality than MSE.

This gives us a reason to study the structural similarity (SSIM) index and related error for measuring self-similarity.

5.1 Introduction

The Structural Similarity Index has been mentioned in the chapter "Mathematical Background". Different from MSE, the SSIM Index mainly aims to consider structural features of images such as blurriness, noisiness, and blockiness. In addition, the core advantage is that SSIM is highly adapted to the human visual system by taking the separation of structural information from images.

The SSIM Index determines differences between luminance, contrasts, and structures, which leads the overall function to be a combination of the measuring function of these three quantities by multiplying them together. Given two N-dimensional signals x and y, the SSIM function is defined as

$$S(x,y) = \frac{2\bar{x}\bar{y} + C_1}{\bar{x}^2 + \bar{y}^2 + C_1} \frac{2s_x s_y + C_2}{s_x^2 + s_y + C_2} \frac{s_{xy} + C_3}{s_x s_y + C_3}, \tag{21}$$

where the parameters C_1, C_2, C_3 are small positive constants and

$$\bar{u} = \frac{1}{N} \sum_{i=1}^{N} u_i,$$

$$s_{uv} = \frac{1}{N-} \sum_{i=1}^{N} (u_i - \bar{u})(v_i - \bar{v}),$$

$$s_u^2 = \frac{1}{N-1} \sum_{i=1}^{N} (u_i - \bar{u})^2.$$

The motivation for the definition of the SSIM Index comes from Weber's Law of perception [5].

$S(x, y)$ represents a "similarity" between x and y, $S(x, y)$= 1 if $x = y$, $S(x, y)$= -1 if $x = -y$.

In order to compare SSIM with L^2-based distance functions, we define an SSIM-based "disimilarity" distance function between x and y as follows,

$$T(x, y) = \sqrt{1 - S(x, y)}. \tag{22}$$

Then the"SSIM-based approximation error" between two subblocks will be

$$T(u, v) = \sqrt{1 - S(u, \alpha v + \beta)} \tag{23}$$

in our computation.

Once again, three cases will be considered,

1. Case 1: $\alpha = 1$, $\beta = 0$;

2. Case 2: $\alpha = 1$, $\beta = \bar{u} - \bar{v}$;

3. Case 3: $\alpha = \frac{s_{uv}}{s_v^2}$, $\beta = \bar{u} - \alpha\bar{v}$.

5.2 Case 1 and Case 2

In Case 1, similar with it in L^2 distance, when $\alpha = 1, \beta = 0$, the approximation is $u \approx v$, so the SSIM function becomes

$$S(u, v) = \frac{2\bar{u}\bar{v} + C_1}{\bar{u}^2 + \bar{v}^2 + C_1} \frac{2s_u s_v + C_2}{s_u^2 + s_v + C_2} \frac{s_{uv} + C_3}{s_u s_v + C_3} \tag{24}$$

and the "SSIM-based approximation error" is

$$T(u, v) = \sqrt{1 - S(u, v)}. \tag{25}$$

Using them into the code of MATLAB (APPENDIX B), we obtain the following distribution of errors in terms of SSIM index,

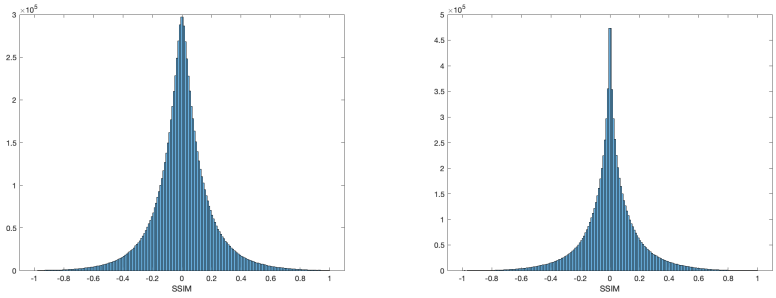


Figure 9: Case 1 SSIM error of *LENA* (left) and *PEPPERS* (right).

and"SSIM-based approximation error" $T = \sqrt{1-S}$:

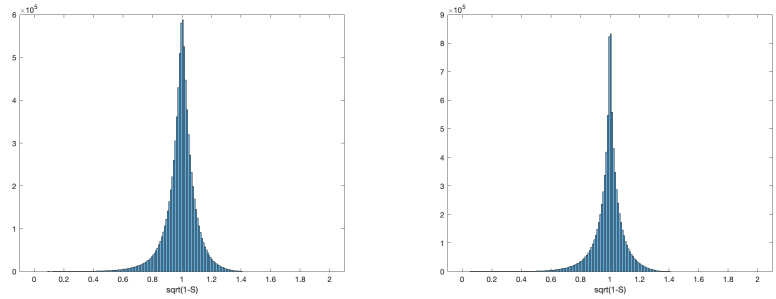


Figure 10: Case 1 $\sqrt{1-S}$ of *LENA* (left) and *PEPPERS* (right).

Similarly, in Case 2 we have approximation $\alpha = 1, \beta = \bar{x} - \bar{y}$, the approximation is $u \approx v + \beta$,

$$S_2(u, v+\beta) = \frac{s_{uv} + C_2}{s_u^2 + s_v^2 + C_2}, \tag{26}$$

the results are shown before.

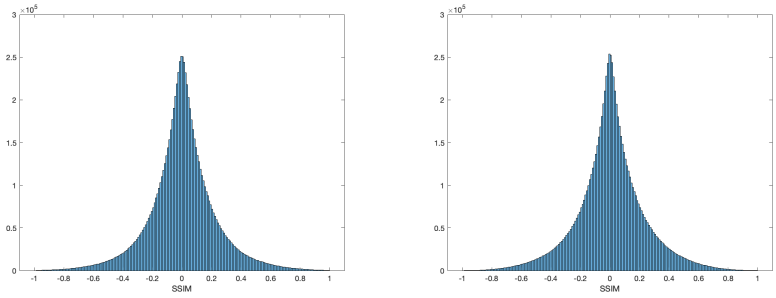


Figure 11: Case 2 SSIM error of *LENA* (left) and *PEPPERS* (right).

"SSIM-based error" $T = \sqrt{1-S}$:

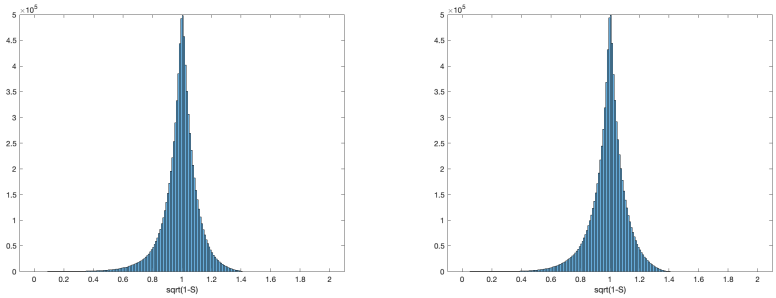


Figure 12: Case 2 $\sqrt{1-S}$ of *LENA* (left) and *PEPPERS* (right).

We can see comparison from Case 1 to Case 2, there is an improvement, but not as significant as it is in L^2 distance. This shows that images are not as self-similar in terms of visual quality (SSIM) as they are in terms of MSE.

5.3 Case 3

In Case 3, the approximation is $u \approx \alpha v + \beta$, there will be two different conditions to define the values of α and β.

Zero Stability Constant

We start with the case of zero stability constants i.e. $C_1 = C_2 = C_3 = 0$. The SSIM function becomes

$$S(x,y) = S_1(x,y)S_2(x,y) = \frac{2\bar{x}\bar{y}}{\bar{x}^2+\bar{y}^2}\frac{2s_{xy}}{s_x^2+s_y}, \tag{27}$$

with $x = u, y = \alpha v + \beta$.

$$S(u, \alpha v+\beta) = S_1(u, \alpha v+\beta)S_2(u, \alpha v+\beta), \tag{28}$$

where

$$S_1(u, \alpha v+\beta) = \frac{2\bar{u}(\alpha\bar{v}+\beta)}{\bar{x}^2+(\alpha\bar{v}+\beta)^2}. \tag{29}$$

Now impose stationarity conditions,

$$\frac{\partial S(u, \alpha v+\beta)}{\partial \alpha} = 0; \quad \frac{\partial S(u, \alpha v+\beta)}{\partial \beta} = 0. \tag{30}$$

We take the component functions separately, then S_1 is

$$S_1(u, \alpha v+\beta) = \frac{2\bar{u}(\alpha\bar{v}+\beta)}{\bar{u}^2+(\alpha\bar{v}+\beta)^2} \tag{31}$$

and S_2

$$S_2(u, \alpha v+\beta) = \frac{2\alpha s_{uv}}{s_u^2+\alpha^2 s_v^2}. \tag{32}$$

Then from previous equations we take derivative of S_1 in terms of β

$$\begin{aligned}\frac{\partial S_1(u, \alpha v+\beta)}{\partial \beta} &= \frac{2\bar{u}[\bar{u}^2-(\alpha\bar{v}+\beta)^2]}{[\bar{u}^2+(\alpha\bar{v}+\beta)^2]^2} \\ &= 0.\end{aligned} \tag{33}$$

In order that S_1=1, the relation between $\bar{u}$ and $\bar{v}$ is as follows,

$$\bar{u} = \alpha\bar{v} + \beta, \quad (\bar{x} = \bar{y}). \tag{34}$$

Similarly, take derivative of S_2 in terms of α

$$\begin{aligned}\frac{\partial S_2(u, \alpha v + \beta)}{\partial \alpha} &= \frac{2s_{uv}}{(s_u^2 + \alpha^2 s_v^2)^2}[s_u^2 - \alpha^2 s_v^2] \\ &= 0.\end{aligned} \tag{35}$$

The equation to obtain the value of α is

$$\alpha^2 = \frac{s_u^2}{s_v^2}. \tag{36}$$

and the final result is

$$\alpha = \pm\frac{s_u}{s_v}. \tag{37}$$

now combine the equation (30) and(33) together, the value of β would be

$$\beta = \bar{u} \mp \frac{s_u}{s_v}\bar{v}. \tag{38}$$

In order to make S_2 >0, we choose

$$\alpha = sgn(s_{uv})\frac{s_u}{s_v}. \tag{39}$$

Earlier, we derived the best L^2 parameter α_{L^2} and β_{L^2}. We now compare them with optimal SSIM parameters

$$\frac{\alpha_{SSIM}}{\alpha_{L^2}} = \frac{\beta_{SSIM} - \bar{u}}{\beta_{L^2} - \bar{u}} = \frac{s_u s_v}{|s_{uv}|}. \tag{40}$$

The next step it to use them in the code (APPENDIX B), then SSIM distances between all pairs of 8X8-pixel blocks $u \approx \alpha v + \beta$ are shown below

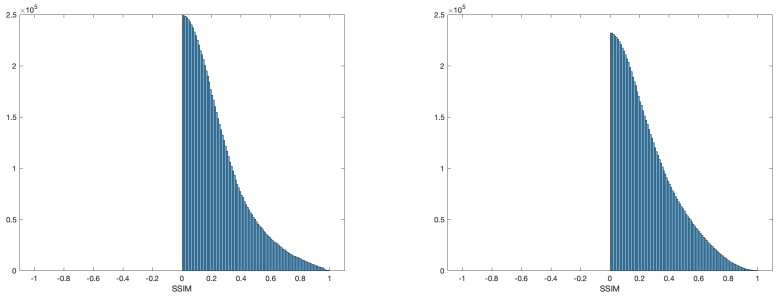


Figure 13: Case 3 with zero stability constant SSIM of *LENA* (left) and *PEPPERS* (right).

The result of $T(u, \alpha v + \beta)$ are

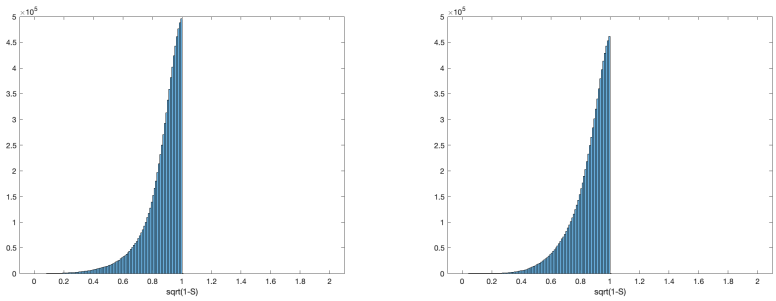


Figure 14: Case 3 with zero stability constant $\sqrt{1-S}$ of *LENA* (left) and *PEPPERS* (right).

We can focus on the figures of function $T(u, \alpha v + \beta)$, compared with Case 1 and Case 2, the histogram in Case 3 is closer to zero because α and β are added, which makes the approximation goes better.

Non-zero Stability Constant

We now Consider non-zero stability constant case. First let $C_3 = 2C_2$, so that the SSIM function simplified as

$$S(x, y) = S_1(x, y)S_2(x, y) = \frac{2\bar{x}\bar{y} + C_1}{\bar{x}^2 + \bar{y}^2 + C_1} \frac{2s_x s_y + C_2}{s_x^2 + s_y + C_2}. \tag{41}$$

Then let $x = u, y = \alpha v + \beta$, so that the equation above becomes

$$S(u, \alpha v + \beta) = S_1(u, \alpha v + \beta)S_2(u, \alpha v + \beta). \tag{42}$$

Now impose stationarity conditions to get the maximum values

$$\frac{\partial S(u, \alpha v + \beta)}{\partial \alpha} = 0; \quad \frac{\partial S(u, \alpha v + \beta)}{\partial \beta} = 0 \tag{43}$$

For convenience, we examine the component functions separately, first from S_1

$$S_1(u, \alpha v + \beta) = \frac{2\bar{u}(\alpha\bar{v} + \beta) + C_1}{\bar{u}^2 + (\alpha\bar{v} + \beta)^2 + C_1}, \tag{44}$$

and for S_2

$$S_2(u, \alpha v + \beta) = \frac{2\alpha s_u s_v + C_2}{s_u^2 + \alpha^2 s_v^2 + C_2}. \tag{45}$$

We can see that S_2 is only related with α, so we only need to take derivative of S_2 with respect to β

$$\begin{aligned}
\frac{\partial S_1(u, \alpha v + \beta)}{\partial \beta} &= \frac{[\bar{u}^2 + (\alpha\bar{v} + \beta) + C_1](2\bar{u}\bar{v}) - [2\bar{u}(\alpha\bar{v} + \beta) + C_1][2(\alpha\bar{v} + \beta)\bar{v}]}{[\bar{u}^2 + (\alpha\bar{v} + \beta)^2 + C_1]^2} \\
&= \frac{2\bar{u}\bar{v}[\bar{u}^2 - (\alpha\bar{v} + \beta)^2] + 2C_1\bar{v}[\bar{u} - (\alpha\bar{v} + \beta)]}{[\bar{u}^2 + (\alpha\bar{v} + \beta)^2]^2} \\
&= 0.
\end{aligned} \tag{46}$$

From the above equation, we have the following result, the stationary condition is

$$\bar{u} = \alpha\bar{v} + \beta, \tag{47}$$

In which case S_1 will be 1. This result hasn't change as compared with the zero stability constants condition, which means that the relationship between $\bar{u}$ and $\bar{v}$ is independent on the constants.

If we take α=1, then $\bar{u} = \bar{v} + \beta$, this is the Case 2 approximation. For S_2 we have

$$\begin{aligned}
\frac{\partial S_2(u, \alpha v + \alpha)}{\partial \beta} &= \frac{2s_{uv}[s_u^2 - \alpha^2 s_v^2] + 2C_2[s_{uv} - \alpha s_v^2]}{[s_u^2 + \alpha^2 s_v^2 + C_2]^2} \\
&= 0.
\end{aligned} \tag{48}$$

(since S_2 is not dependent on β, there is no need to take derivative with regard to β).This equation is zero when

$$s_{uv}[s_u^2 - \alpha^2 s_v^2] + C_2[s_{uv} - \alpha s_v^2] = 0, \tag{49}$$

which is quadratic equation in α,

$$\begin{aligned} s_{uv}s_v^2\alpha^2 + C_2 s_v^2\alpha - s_{uv}[s_u^2 + C_2] = 0, \\ \alpha^2 + \left(\frac{C_2}{s_u v}\right) - \frac{1}{s_v^2}[s_u^2 + C_2] = 0. \end{aligned} \tag{50}$$

This equation has two solutions,

$$\alpha_\pm = -\frac{C_2}{2s_{uv}} \pm \left[(\frac{C_2}{s_{uv}})^2 + \frac{4}{s_v^2}[s_u^2 + C_2]\right]^{\frac{1}{2}}. \tag{51}$$

This is a new result for the case of nonzero stability constant.

If we take the limit $C_2 \to 0$, we have the limit $\alpha \to \pm\frac{s_u}{s_v}$, which is in agreement with the result in zero stability constant condition. This conclusion is desirable.

Again, the figures of non-zero stability constants condition are

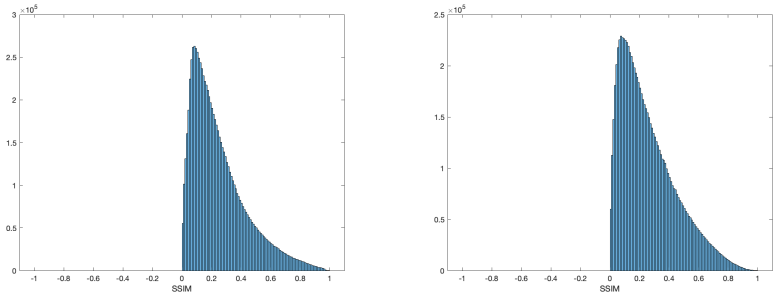


Figure 15: Case 3 with non-zero stability constant SSIM of *LENA* (left) and *PEPPERS* (right). $C_1 = C_2 = 0.00001$.

The figures for the function T are

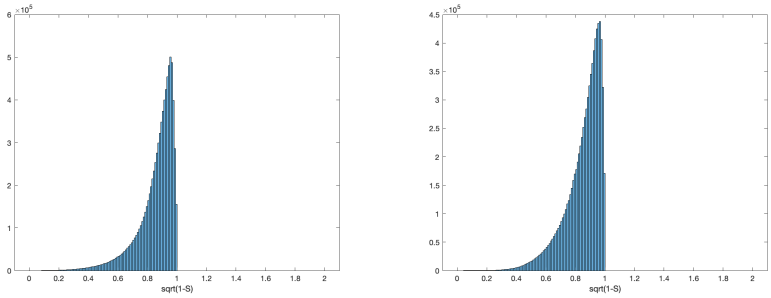


Figure 16: Case 3 with non-zero stability constant $\sqrt{1-S}$ of *LENA* (left) and *PEPPERS* (right). $C_1 = C_2 = 0.00001$.

Comparison between zero and non-zero conditions is obvious:

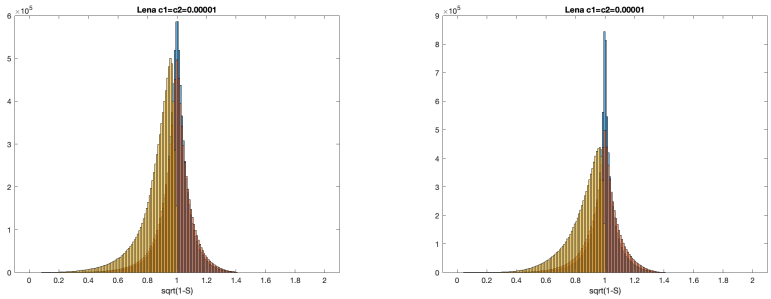


Figure 17: Case 1,2,3 with non-zero stability constant $\sqrt{1-S}$ of *LENA* (left) and *PEPPERS* (right). $C_1 = C_2 = 0.00001$.

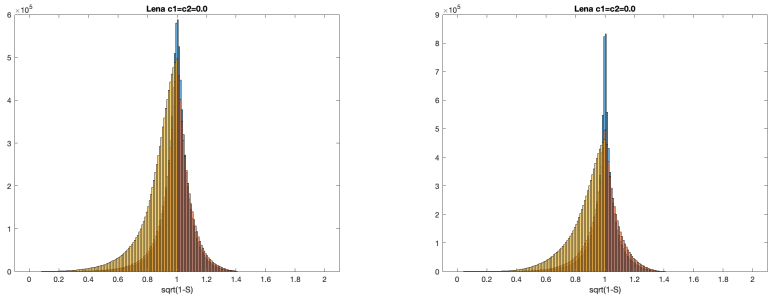


Figure 18: Case 1,2,3 $\sqrt{1-S}$ of *LENA* (left) and *PEPPERS* (right) with zero constants

Recall that when we use L^2 distance to calculate self-similarity, there is

significant improvement from Case 1 to Case 2. This is because it is easier to approximate low-variance subblocks. However, SSIM-based self-similarity is not so distinct between different cases.

Let me explain this briefly. It is well known that the best constant approximation of an image subblock u is the mean of the subblock, i.e. $u \approx \bar{u}$. To see this, we minimize the squared L^2 error

$$\Delta = \sum_{i=1}^{N} (u_i - c)^2. \tag{52}$$

Impose the stationary condition,

$$\frac{\partial \Delta}{\partial c} = -2 \sum_{i=1}^{N} (u_i - c) = 0, \tag{53}$$

which yields

$$c = \frac{1}{N} \sum_{i=1}^{N} u_i = \bar{u}. \tag{54}$$

The squared L^2 error of this approximation is

$$\Delta = \sum_{i=1}^{N} (u_i - \bar{u})^2 = (N-1)\delta_u^2. \tag{55}$$

In other words, the small the variance of u, the lower the error of approximation by a constant.

The best constant approximation of a subblock u using SSIM is also $u \approx \bar{u}$. To see this, the first component of the SSIM index, $S_1(u, c)$,

$$S_1(u, c) = \frac{\bar{u}c}{\bar{u}^2 + c^2}, \tag{56}$$

is maximized when $c = \bar{u}$, in which case $S_1(u, \bar{u})$=1. However,

$$S_2(u, c) = \frac{2s_{uc}}{s_u^2 + 0} = \frac{2s_{uc}}{s_u^2} \tag{57}$$

and

$$s_{uc} = \frac{1}{N-1} \sum_{i=1}^{N} (u_i - \bar{u})(\bar{c} - \bar{c}) = 0. \tag{58}$$

Therefore

$$S(u, \bar{u}) = 0 \tag{59}$$

In other words, the best SSIM-based constant approximation to an image subblock is also the mean of u, $\bar{u}$, but its SSIM value is always 0. This implies that $T(u, \bar{u})$=1. It can never be improved.

This shows that images are not as self-similar in terms of visual quality (SSIM) as they are in term of MSE.

6 Conclusions

In our research, we revisited self-similarity in terms of two measures based on pixel domain; inspired by the fact that many imaging techniques are based on self-similarity. We started from images represented by two-dimensional functions, then introduced two common distances for calculation, MSE and SSIM respectively.

MSE is one of the most widely used method in calculating self-similarity, based on L^2 distance. The main idea is making an approximation of $u \approx \alpha v + \beta$, working on 3 cases with different α and β to see how they influence the error. In comparison of histograms for Case 1, 2, a significant improvement to zero are shown. However, the weakness is that MSE is not a good measure of visual quality, since low MSE does not necessarily mean good quality. In order resolve this problem, SSIM, which is a is a better measure of image qualities, introduced after. Following similar process to make an approximation, same conclusion for both MSE and SSIM would be stated here, Case 2 errors with one parameter are smaller than Case 1 errors with no parameters and Case 3 errors of two parameters are smaller than Case 2 errors. As improvements in term of SSIM are much smaller than improvements for MSE, images are not as self-similar in terms of visual quality as they are in term of L^2 distance.

For the further research, instead of pixel only, another domain called wavelet domain will be introduced later. The wavelet representation of the image is a common setting for image processing. To deal with wavelet, a new function, Harr wavelet will be mentioned, and after we may combine the MSE and SSIM with wavelet. In addition, we will continue to work on some applications of self-similarity, mainly focusing on non-local means denoising technique in image processing. It is achieved by calculating the estimated value of the denoised pixels as a weighted sum of the other pixels in the noisy digital image using feedbacks from different parts from a noisy image, is called “non-local. This is our preliminary decision on what to do next.

7 Acknowledgements

I would like to express my deep and sincere gratitude to my supervisor, Edward. R. Vrscay in Applied Math faculty, University of Waterloo, for his guidance through each stage of the process. Prof. Vrscay is a very kind and patient professor, timely helps were always given by him when I ran into difficulties that I could not solve by myself. It's one of the most precious experiences during my life to have him as my supervisor.

I also want to acknowledge Dongchang Li, the graduate student of Prof. Vrscay, who gave me much help in using LateX and writing this report. In addition, his useful advices benefited me a lot.

Last of all, I'm grateful for my parents, who have paid for my tuition fees in University of Waterloo and always supported me whatever I decided to do.

8 APPENDICES

8.1 APPENDIX A

```
A = imread('lenna.pgm');%A=imread('pepper.pgm');
A = uint8(A);
npix=8;
sizeA = size(A);
N=npix*npix;
NA = (sizeA(1)/npix)*(sizeA(2)/npix);
for i=1:sizeA(1)/npix
for j=1:sizeA(2)/npix
sum1=0.0;
for k=1:npix
for l=1:npix
A8(k,l)=double(A((i-1)*npix+k,(j-1)*npix+l))/255.0;
sum1=sum1+A8(k,l);
end
end
amean(i,j)=sum1/N;
sum1=0;
for k=1:npix
for l=1:npix
sum1=sum1+(A8(k,l)-amean(i,j))
end
end
avar(i,j)=sum1/(N-1);
end
end
kk=0;
ia=0;
for i1=1:sizeA(1)/npix
for j1=1:sizeA(2)/npix
ia=ia+1;
for k=1:npix
for l=1:npix
A8(k,l)=double(A((i1-1)*npix+k,(j1-1)*npix+l))/255.0;
end
end
ib=0;
```

```
for i2=1:sizeA(1)/npix
for j2=1:sizeA(2)/npix
 ib=ib+1;
if (ib>ia)
kk=kk+1;
for m=1:npix
for n=1:npix
B8(m,n)=double(A((i2-1)*npix+m,(j2-1)*npix+n))/255.0;
end
end
sum2=0.0;
sum3=0.0;
for m=1:npix
for n=1:npix
sum2=sum2+(A8(m,n)-amean(i1,j1))*(B8(m,n)-amean(i2,j2)
    );
sum3=sum3+(A8(m,n)-B8(m,n))*(A8(m,n)-B8(m,n));
end
end
acov=sum2/(N-1);
tsqerr1=sum3;
mse1=tsqerr1/N;
rmse1=sqrt(mse1);
case1(kk)=rmse1;
tsqerr2=(N-1)*avar(i1,j1)+(N-1)*avar(i2,j2)-2*(N-1)*
    acov;
mse2=tsqerr2/N;
rmse2=sqrt(mse2);
case2(kk)=rmse2;
tsqerr3=(N-1)*avar(i1,j1)-(N-1)*(acov/avar(i2,j2);
mse3=tsqerr3/N;
rmse3=sqrt(mse3);
case3(kk)=rmse3;
end
end
end
end
end
figure(1)
imshow(A); %original image;
```

```
figure(2)
histogram(case1,100),xlabel('RMSE case1');%Case1
figure(3)
histogram(case2,100),xlabel('RMSE case2');%Case2
figure(4)
histogram(case3,100),xlabel('RMSE case3');%Case3
```

8.2 APPENDIX B

```
A = imread('lenna.pgm');
%A = imread('peppers.pgm');
A = uint8(A);
sizeA = size(A);
bitsA = sizeA(1)*sizeA(2)*8;
NA = sizeA(1)*sizeA(2);
npix=8;
np=npix*npix;
c1=0.0;
c2=0.0;
%c1=0.00001;
%c2=0.00001;
for i=1:sizeA(1)/npix,
for j=1:sizeA(2)/npix,
A8=A((i-1)*npix +1:i*npix,(j-1)*npix+1:j*npix);
A8=double(A8)/255.0;
amean(i,j)=mean(A8(:));
avar=var(A8(:));
saa(i,j)=avar;
end;
end;
kk=0;
ia=0;
for i=1:sizeA(1)/npix,
for j=1:sizeA(2)/npix,
ia=ia+1;
A8=A((i-1)*npix +1:i*npix,(j-1)*npix+1:j*npix);
B8=double(A8)/255.0;
ib=0;
for k=1:sizeA(1)/npix,
for l=1:sizeA(2)/npix,
```

```
ib=ib+1;
if (ib > ia)
kk=kk+1;
B8=A((k-1)*npix +1:k*npix,(l-1)*npix+1:l*npix);
B8=double(B8)/255.0;
sab = sum(sum( (double(A8)-amean(i,j)*ones(npix)).*(
    double(B8)-amean(k,l)*ones(npix))))/double(np-1);
case1a = (2.0*amean(i,j)*amean(k,l)+c1)/(amean(i,j)
    amean(k,l)c1);
case1b = (2.0*sab+c2)/(saa(i,j)+saa(k,l)+c2);
case1(kk) = case1a*case1b;
case2(kk) = case1b;
t1=(c2/sab) + 4.0*(saa(i,j)+c2)/saa(k,l);
if (sab < 0) alpha=-0.5*c2/sab - 0.5*sqrt(t1);
end
if (sab > 0) alpha=-0.5*c2/sab + 0.5*sqrt(t1);
end
case3(kk) = (2.0*alpha*sab+c2)/(saa(i,j)+alphasaa(k,l)
    +c2);
end;
end;
end;
end;
end;
kmax=kk;
t1=sqrt(1.0-case1);
t2=sqrt(1.0-case2);
t3=sqrt(1.0-case3);

figure(1)
imshow(A);

figure(2)
histogram(case1,200,'BinLimits',[-1,1]),xlabel('SSIM')
    ,title('Lena c1=c2=0.0');
histogram(case1,200,'BinLimits',[-1,1]),xlabel('SSIM')
    ,title('Lena c1=c2=0.00001');
hold on
histogram(case2,200,'BinLimits',[-1,1]),xlabel('SSIM')
    ;
```

```
hold on
histogram(case3,200,'BinLimits',[-1,1]),xlabel('SSIM')
    ;


figure(3)
histogram(t1,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)'
    ),title('Lena c1=c2=0.0');
histogram(t1,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)'
    ),title('Lena c1=c2=0.00001');
hold on
histogram(t2,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)'
    );
hold on
histogram(t3,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)'
    );

figure(4)
histogram(case3,200,'BinLimits',[-1,1]),xlabel('SSIM')
    ;
%histogram(case2,200,'BinLimits',[-1,1]),xlabel('SSIM
    ');
%histogram(case1,200,'BinLimits',[-1,1]),xlabel('SSIM
    ');

figure(5)
histogram(t3,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)'
    );
%histogram(t2,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)
    ');
%histogram(t1,200,'BinLimits',[0,2]),xlabel('sqrt(1-S)
    ');
```

9 References

[1] D. Glew. (2011). Self-Similarity of Images and Non-local Image Processing. UWSpace. http://hdl.handle.net/10012/6019

[2] D. Brunet, E.R. Vrscay and Z Wang. (2011). Structural Similarity-Based Affine Approximation and Self-similarity of Images Revisited. In: Kamel M.,Campilho A. (eds) Image Analysis and Recognition. ICIAR 2011. Lecture Notes in Computer Science, vol 6754. Springer, Berlin, Heidelberg

[3] S. Alexander, E.R. Vrscay, S. Tsurumi. A Simple, General Model for the Affine Self-similarity of Images. DOI: 10.1007/978-3-540-69812-$8_1$9

[4] H. Kunze, D. La Torre, F. Mendivil, E.R. Vrscay. Fractal-Based Methods in Analysis Springer. ISBN 978-1-4614-1891-7

[5] Z Wang, A.C. Bovik. (2018). Mean squared error: Love it or leave it? A new look at Signal Fidelity Measures.IEEE Signal Processing Magazine (Volume:26, Issue:1, Jan. 2009)