”ESTIMATION AND COMPARISON OF THE IMAGE NOISE LEVELS VIA SUBSAMPLING”

A Thesis

by

DUNG ANH LE

MS, Texas A&M University-Corpus Christi, 2019

Submitted in Partial Fulfillment of the Requirements for the Degree of

MASTER OF SCIENCE

in

MATHEMATICS

Texas A&M University-Corpus Christi
Corpus Christi, Texas

August 2019

August 2019

”ESTIMATION AND COMPARISON OF THE IMAGE NOISE LEVELS VIA SUBSAMPLING”

A Thesis

by

DUNG ANH LE

This thesis meets the standards for scope and quality of
Texas A&M University-Corpus Christi and is hereby approved.

Dr. Lei Jin, PhD
Chair

Dr. Blair Sterba-Boatwright, PhD
Co-Chair

Dr. Yuxia Huang, PhD
Committee Member

August 2019

ABSTRACT

As the amount of digital data has increased critically in the last decade, image data has become more and more important. Noise is a random signal which always present in an image during image acquisition, coding and, transmission. Image noise leads to pixels representing incorrectly the color or the exposure of the scene. The noise level is an important component for measuring the quality of an image.

In the literature, very little work has been done in statistical inference for image noise. Most existing methods deal with the point estimation of the noise level, but the sampling distribution of these point estimates are unknown in general. In this project, we propose sub-sampling methods for image data to approximate the sampling distribution for the point estimates. Also, we develop some methods to compare the noise level of two images. First, we review different models of image noise including independent noise, dependent noise and bivariate noise models. Usually the probability models for image noise are not simple and the variance estimates are complicated. The estimates themselves require sampling distributions for statistical inference. Second, we approximate the sampling distributions via subsampling methods. In statistics, bootstrap and resampling methods are widely used to construct the sampling distributions and estimate the variances of different statistics. Here, we generalize resampling methods for one dimensional data to deal with two dimensional images. From these, confidence intervals for the noise level are constructed. Also, some methods for comparing the image variance in paired images are evaluated for different types of noise such as independent noise and dependent bivariate Gaussian noise. The results of the estimation noise level and hypothesis tests on variance comparison are provided. It seems that the proposed subsampling methods provide reasonable results while both the F-test and the Pitman t-test may not work well.

ACKNOWLEDGEMENTS

I would like to acknowledge the support of the Department of Mathematics and Statistics and the College of Graduate Studies at TexasAMUniversity-Corpus Christi. Their wonderful supports by the time I work on my degree is out of my expectation. Also, I would like to send my gratefulness to my committee members who are very enthusiastic to support for my thesis. Especially, I want to express my deepest appreciation to Dr. Lei Jin, the chair of my committee. He gave me his fantastic patience when advising me to my goal. I am extremely grateful to Dr. Blair Sterba-Boatwright who always gave me warm advice whenever I need them during the time I study at the Department of Mathematics Statistics.

TABLE OF CONTENTS

LIST OF FIGURES

FIGURES PAGE

CHAPTER I: INTRODUCTION

In the last decade, the digital data increases critically due to the development of technology and social media. Image data is one of the most important categories of digital data and are used widely for different purposes. For example, the metadata of a computed tomography scan might give doctors a signal of a decease, or scientists might classify images from satellites to determine the expansion of oil leaks on the ocean surface. However, working on image data is more difficult than regular data because the tools for statistical inference are lacking.

Usually, images are captured mostly from digital camera or scanned from other resources. The pixel is the smallest element of a digital image and it represents the color at a specific place on the image depending on its value. For instance, the minimum value of 0 represents a black dot on most type of image. The value of a pixel might have different range due to the type of image and it corresponds to the light intensity at that point [1].

People classify the image data by using different methods. Depending on the color channels, digital images may be black and white, grayscale or RGB (Red, Green and Blue). These types of images can be converted from one channel to other color bases using different models. Each color channel has its own advantages and disadvantages when we want to process the image. For instance, a color image gives better visualization but takes longer processing time compared to a grayscale image. By processing the color channels of the image, we can obtain important hidden information depending on different color bands.

About the content of the picture, digital images are not only the description of the objects but also contain more important information such as the features of the image and the metadata. For example, x-ray images obtained from a different beams carry various properties after complicated analysis [2]. By extracting the metadata from the set, we can obtain relevant details from the image itself. For example, the object inside an image can be retrieved by different algorithms to determine which part on the image might contain an object. Also, there are other ways to classify the image depending on the purpose of users.

Another important part of image data is the image handler which includes tools and libraries of programming language. When the image data was classified by different point of view, it will need several handlers such as ”CImg” class in R[3]. This object belongs to ”Imager” library which contains a large array of functions for working with image data. Imager is based on CImg library, a C++ open sourced library developed by David Tschumperl. The CImg provides an easy-to-use and consistent Application Programming Interface (API) for image processing when the image is largely replicated. With CImg, the user can load or save various file format. Also, the most important part of CImg is to compute statistics and generate noises model before applying the noise onto the image.

Image quality is an important property in some fields such as medical or logistics. The only possible method to estimate the image quality is to compare it subjectively to the original. The amount of image data is increasing much too fast to permit widespread assessment of image quality by humans. However, there is no accepted standard to assess image quality [4]. Image noise (defined next section) can affect the quality of an image seriously. Therefore, we need to find a method to quantify image noise to assess image quality mechanically.

1.1 Image Noise

Once a digital image is captured, there are multiple treatments until it is displayed. These processes include capturing, compression, transferring, decompression, storing and displaying. These complicated steps lead to the degrading of the image quality. Some effects on the picture are blur, reduced texture, low light or too bright images. These effects are also applied on the analog printed photograph due to the failure of the device. In general, we call these effects image noise.

Basically, image noise consists of additional variables that produce the effect of random variation of brightness or color. They are applied on the real values of each pixel. In most captured images, these unwanted signals are unavoidable. The reason can come from the photographer or the devices which are used to take the picture (like cameras, scanners or sensors) or from the system which processes the image (compression and decompression) before it is displayed. Since noise is inevitable, this issue becomes one of the most important part of image processing.

Noise level is an important parameter to multiple image processing systems such as classification,

segmentation or denoising. For the image assessment process, we can try to estimate the noise level of each image. Then, we can apply statistical tests to compare the image noise levels of different images.

1.2 Types of image noise

There are multiple models for the types of noise that affect an image. One of the most popular is Gaussian Noise which may be caused by natural resources such as poor illumination, high temperature or transmission of the digital image. The effects on the image with Gaussian noise are dark or bright areas which can highly reduce the quality of the picture. This type of noise is called statistical noise when the additive noise has the probability density function of normal distribution.

Another popular noise model is called salt and pepper or spike. We can recognize this type of noise through black and white pixels on the image. The reason of this noise comes from the disturbance of the image signal. The value of pixels is 0 or 255 resulting in a black or white dot. There are many other types of noise such as anisotropic noise (row or column noise), quantization noise (uniform noise), periodic noise, etc.

For many applications of image data, noise reduction is necessary to improve the accuracy of the application. The noise can be removed by using some special filters. However, the types of filter used varies depending on the type of image noise that needs to be removed. Before we can perform the process to remove the image noise, we need to determine the noise level of the image. Depending on the noise level, we can decide which filter can be used. This parameter is important for image denoising model, image segmentation or resolution improvement systems [5]. As a result, estimating the noise level of the image accurately is challenging when the image has a complex texture.

1.2.1 Noise level of images

In many proposed models, the authors quantify image noise as the variance of the image [6][7]. When each image is a description of an object, the structure of each image pixel is affected differently by the same type of noise. For instance, anisotropic noise makes the edge of the image becoming harder

to detect. Therefore, we need an effective method to calculate the standard deviation or the variance of the image instead of using the regular statistical formula.

1.2.2 Comparing the noise levels of two images

There are multiple situations that call for the comparison of two images. For example, we need to compare pictures a person's face to train a facial recognition system. Alternatively, we may need to determine the vehicle plates through the cameras on toll roads at different locations for a lawsuit. Some situations provide good quality images that we can use to determine if the same object appears on multiple pictures. However, many situations give us noisy images when the cameras do not operate under the best conditions. The solution can be improving the image quality and the use of regular object detecting applications to compare the whole dataset. Nevertheless, the noise removing process costs much time and effort when the collected data is large. Also, sometimes it requires human effort to determine how close those pictures are. In this case, we can use a statistical method to compare those images if the effort to calculate the variance of the image and performing hypothesis tests for comparison have lower time costs. In addition, the result can be determined if the hypothesis claim to compare two image variances is true or not.

We also can consider sub-sampling techniques when we want to compare two images. In image processing, sub-sampling is used to take a part of the image by reducing its dimension to get smaller images. In statistic field, sub-sampling is a technique that help us to determine the distribution type of the data. By performing statistical sub-sampling method on the image, we can determine the statistic numbers of the image. Then, we can compare the noise level of two images.

1.2.3 How the thesis is organized

In this paper, chapter two describes some methods that we can use to determine the variance of the image such as Weak Texture Patches or subsampling. Then, chapter three will provide information of statistical tests that we can use to compare images of the same subject. A proposed method to compare image variance using subsampling is also provided. Chapter four gives some results of these

tests applied on an image dataset that contains different noisy images of the same picture. Then, some conclusions are made in chapter five.

CHAPTER II: ESTIMATION OF THE NOISE LEVELS

In statistics, variance is the expectation of the squared deviation of a random variable from its mean. To calculate the variation within the pixels, we can transform image pixels into a vector which can be considered as data in Euclidean space. To this vector, we can apply the standard definition of a sample variance:

$$\sigma = \sum_{i=1}^{N} \frac{(x_i - \bar{x})^2}{N-1}$$

where $\bar{x}$ is the sample mean of the image. However, the variation across pixels is more important to determine the noise level of the image. In addition, two images might have the same variation at the same noise level. Therefore, we need a different method to measure the variance of the noisy image through some models. Due to the relationship between the noise itself and the image, the structure of the image is important when we care about the noise. When pixels are unrelated to each other, they still create some texture in a small partition of the image such as edges. Therefore, we cannot use this concept when we want to calculate the variance of the image. We can assume that the noise level is independent to the image itself. As a result, each value of the noise level is statistical independent. This chapter will describe some possible methods that estimate the image variance through the image structure.

Several methods for image noise estimation have been proposed. Ce Liu introduced a method to estimate the image noise level from just a single image [6]. Liu, Tanaka and Okutomi estimated the dependent noise on an image [8]. Principal Component Analysis is also used to determine the weak texture patches before calculating the image noise [7]. The detail of these methods will be mentioned in this chapter,

2.1 Noise level estimation with Weak Texture Patches (WTP)

This subsection reviews Principle Component Analysis and how other people have applied PCA on Weak Texture Patches to estimate the noise level.

Many algorithms are proposed for image noise estimation, but they are classified into patch-based or filter-based method. The problem of filter-based methods is their assumption that defines the noise as the difference between the noisy image and the original image. Since some types of noise depend on the structure of the image, this assumption seems to be incorrect. On the other hand, the problem of patch-based method is the selection of weak texture patches after an image is split into patches. A *weak texture patch* is a smoother subset of the entire image which has a low noise level. The texture strength parameter which is defined by [9] can help us to find the weak texture patches. The weak texture patches use PCA and the texture strength metric to estimate the noise level of the image [7].

Principle Component Analysis (PCA) is a statistical procedure that reduces the dimension of a large data set into a set of smaller dimension that still contains the main information of the original set. Mathematically, this method performs orthogonal linear transformation (projection) on the data and repeats the process until the principle components are found. PCA works on a squared symmetric matrix such as covariance matrix or correlation matrix.

The calculation is applied separately on each color channel. First, we can separate each color channel of the big picture into image patches y with sizes much smaller than the original image. These image patches' size can be as small as the filter that we use for the next step. Zhu and Milanfar [10] show that the we can measure image structure effectively by using the corresponding gradient covariance matrix. The Cov_y of the patch y that belong to image size $N \times N$ is defined below[10][7].

$$Cov_y = \Delta_y^T \Delta_y$$

Δ_y which is the gradient matrix over $n \times n$ window with $n < N$ is defined as following:

$$\Delta_y = [D_h y \quad D_v y]$$

D_h and D_v are horizontal and vertical matrix filters on the image patch y. An example of the filter is as below [10].

$$D_h = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad D_c = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Since the image patch information reflects mostly through its eigenvalue and eigenvector, we can apply the singular value decomposition (SVD) for Δ_y, such as

$$\Delta_y = USV^T$$

U and V^T are orthogonal matrices in which columns of U and V are singular vectors. These vectors are the eigenvectors of $\Delta_y\Delta_y^T$ or $\Delta_y^T\Delta_y$[11]. S is a diagonal matrix which has decreasing singular value. Then,

$$Cov_y = \Delta_y^T\Delta_y = VS^TSV^T = V\begin{bmatrix} s_1^2 & 0 \\ 0 & s_2^2 \end{bmatrix}V^T$$

The gradient's strength is related closely to the singular value through the maximum eigenvalue s_1 of the covariance matrix. Therefore, the texture strength of the image patch can be calculated by using this parameter as quantitative measurement. When the eigenvalue is sensitive to the noise, we need to figure out how independent noise affects this value of the gradient covariance matrix. At this point, we will consider a perfectly flat patch size N which contains all points sharing the same intensity value. Since the perfectly flat patch just contains a constant intensity, its gradient should be zero and we can calculate the expected gradient covariance matrix as

$$E(Cov_y) = E(Cov_n) = \begin{bmatrix} E(n^TD_h^TD_hn) & 0 \\ 0 & E(n^TD_v^TD_vn) \end{bmatrix}$$

We will find the moment generating function (MGF) of $k(n) = n^TD_h^TD_hn$ and use the gamma distribution to simplify the problem. Comparing the MGF of $k(n)$ and the gamma distribution, we can approximate the parameters of the gamma distribution as following[7]

$$\alpha = \frac{N}{2}, \beta = \frac{2}{N}\sigma_n^2 tr(D_h^TD_h)$$

Since the range of the noise level does not have a limited range, we can determine a threshold to estimate the noise level. The maximum eigenvalue of the gradient covariance matrix is smaller than some threshold defined as

$$T = \sigma_n^2 G^{-1}(\delta, \frac{N}{2}, \frac{2}{N}\sigma_n^2 tr(D_h^TD_h))$$

where $G^{-1}$is the inverse gamma distribution function with the shape parameter $\frac{N}{2}$ and scale parameter $\frac{2}{N}\sigma_n^2 tr(D_h^T D_h))$. An iteration will make the weak texture patch selected from the noisy image with the correct noise levels. The calculation steps are given in [7].

2.2 Estimation noise level of dependent image noise

Frequently, image noise varies with image texture, so another type of noise model is dependent noise, where the noise level will rely on the structure of the image.

We can still adopt the concept of weak texture selection above to calculate the noise level of images with dependent noise. The gradient covariance matrix is still the same. However, the initial threshold is defined as following:

$$\tau = \sigma_n^2 F^{-1}(\delta, \frac{N}{2}, \frac{2}{N}\sigma_n^2 tr(D_h^T D_h + D_v^T D_v))$$

The process to estimate the noise level of an image with dependent noise is still the same as the weak texture patch method. The initiate threshold is calculated. The necessary patches that will be used to estimate the noise level are selected. Then, the noise level will be estimated.

2.3 The sampling distributions of the noise level estimation

To approximate the distribution of the noise level, we can use bootstrapping or sub-sampling techniques.

2.3.1 Bootstrapping technique

The bootstrap is a statistical method which relies on random resampling of a given sample to estimate the sampling distribution. We can apply this method on image data to estimate the variance of an image by combining it with WTP.

For image data, the resampled image will have the same size as the original image. The resampled image is constructed from multiple rectangular subimages that are collected randomly from the root image and stitched together (see Figure 2.1). The subimages are not expected to be disjoint; that is,

(a) Original Image

(b) Resample built from 100 $\times$ 100 pixels subimages

(c) Resample built from 50 $\times$ 50 pixels subimages

Figure 2.1: A sample of boostrapping applied to image data

different regions of the original image may be included in the various subimages that comprise the resample.

Once a subimage is generated, statistics such as mean and variance are calculated. Then, the whole process is repeated for a large number of resampled images. The result will be a collection of means and variances that can be used as an estimate of the sampling distribution of the mean and variance.

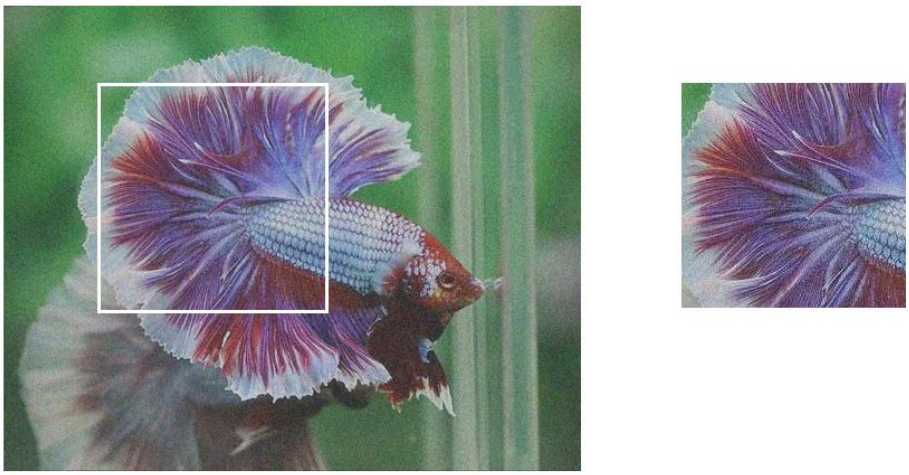
2.3.2 Sub-sampling method

Since the standard calculation of variance based on a whole image is unreliable, we will try to find a range of values for the unknown ”true” variance of the image. We can consider an image is a collection of sub-images. Therefore, the variance of the whole picture can be measured through its fragments. If we can estimate the variance of each sub-image, we will derive an estimated range for the variance of the whole image.

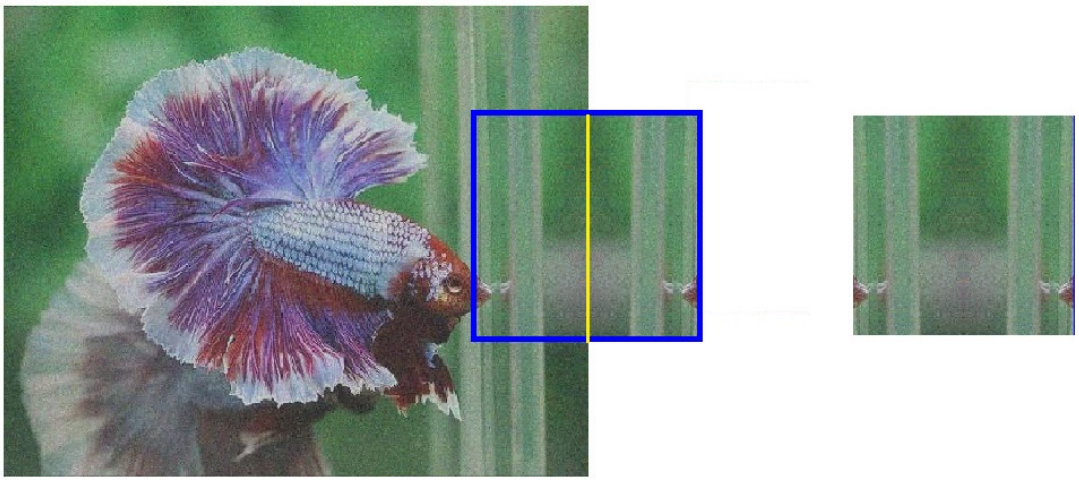
In this method, multiple sub-images are generated from a noisy image. The size of a sample has to be smaller than the size of the noisy picture. The location of the sub-image is picked randomly. Examples of sub-images are shown in figure 2.2.

Sub-images have a fixed size and their locations which are top left corner of the subimages are picked randomly from the root image. To avoid the problems when picking subimages at the edge or corner of the image, distance between the edge and the picked location has to equal to size of the subimage. The figure 2.2a shows a sub-image that was picked randomly near the center the original picture. However, there will be the cases that a sub-image location is picked close to an image border. To complete the sub-image, we may repeat the part that stays inside of the image, or we can control the picking location to avoid this problem. Figures 2.2b and 2.2c give examples of completing a sub-image when it does not stay completely inside of the root image.

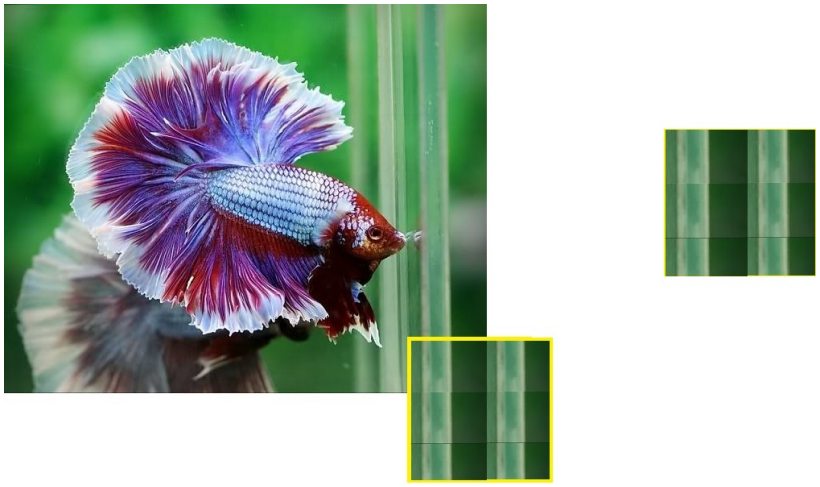
After a sub-image is generated, we can compute the sub-image noise level by applying the weak texture algorithm above. Once the result of one sample is found, the process will be repeated multiple times for different sub-images (about 500 to 1000 samples) with the same size. Then, we can collect all the noise level and estimate the range of the original image variance. We can use this method to compare the variance of two images of the same subject by using a paired subsampling method that is proposed in section 4.5.



(a) Random sub-image fitting the root image



(b) Random sub-image without fitting the root image on edge



(c) Random sub-image without fitting the root image at corner

Figure 2.2: Sub-images of size of 200×200 from different locations of the original image

CHAPTER III: IMAGE NOISE COMPARISON METHODS

There are some situations that comparisons of variance for two images of the same subject are important. For example, in quality control problems after image manipulation, people may want to choose the method with smaller variability. Comparison of the variances can help support their decision making [12]. There are multiple methods that we can use to compare two images through their variances. In this Chapter, we describe two in detail.

3.1 Image residual

If we know the original image, we can compare both noisy images to the original image to find out which one is better. The residual image is defined by the difference between the noise image and the original image. By computing the variances of the residual image, we can see which is truer to the original. However, the root images are not available in most of the cases. As a result, we need to find a different method to determine the better version of the two noisy images.

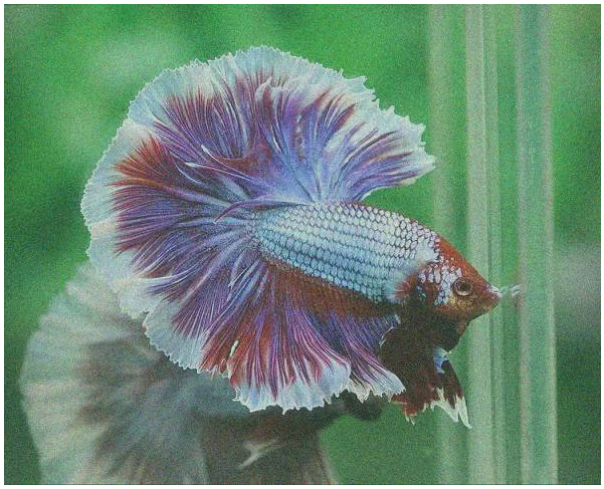
In the technique of image improvement, a smooth image is considered as a better solution from the noisy image. Although the smoothed image cannot describe perfectly the objects inside the image, the added noise is reduced through applying filters. There are a few methods that can help us to smooth the image and the next section will provide more details about them.

3.1.1 Smoothing Images

There are many different ways to obtain a smooth image through blurring technique. In general, each method applies a different filter. For example, the Gaussian blur uses a function for its filter. The formula below is used for one dimensions[13].

$$f(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{\frac{-x^2}{2\sigma^2}}$$

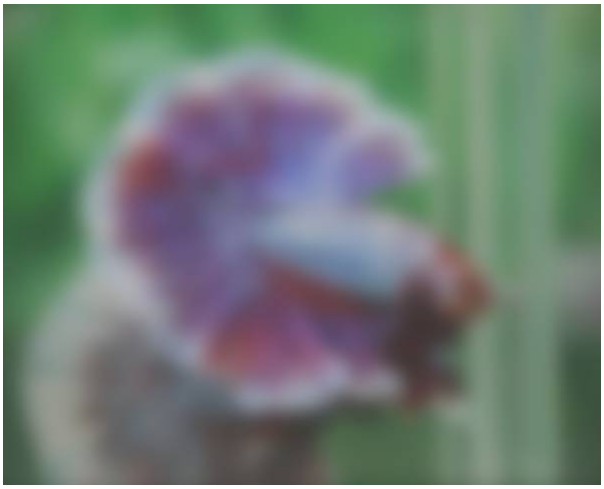
where x is the image pixel and σ is the blur standard deviation. Multiple blurring filters are built that use this concept. The images in Figure 3.3 are blurred using different standard deviations.



(a) Original Noisy Image

(b) $\sigma = 1$

(c) $\sigma = 5$



(d) $\sigma = 10$

Figure 3.3: Blurred Images with different Standard Deviation

Another smoothing filter which is used widely is based on the normalized filter. The value of the image is calculated using an average filter which is constructed basing on the size of the filter. Below is an example of the average filter size $[3 \times 3]$

$$F = \begin{bmatrix} \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\ \frac{1}{9} & \frac{1}{9} & \frac{1}{9} \end{bmatrix}$$

Another popular blurring filter is called median filter. The median of all pixels under the kernel window is computed and then the central pixel of the filter is replaced with the calculated result.

The Gaussian function is used to obtain the smooth image for this thesis.

3.1.2 Calculating Image Residuals

Since the smooth image is created, we can calculate the residual image. The residual image is generated by subtracting the noisy image by its own smooth image as the formula

$$r(i,j) = y(i,j) - s(i,j)$$

where $r(i,j)$ is the residual of the pixel located at (i,j) on the image while y is the noisy image and s is the smoothing image. Some of the residual images are shown in figure 3.4.

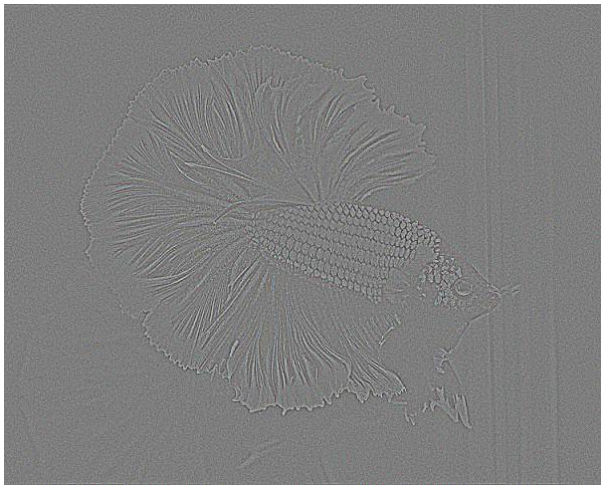
3.2 F-test

We will need a method to compare the image residuals after we calculate them. When both images might have different noise, we can consider each residual in an image. As a result, we need to compare the variance of these two sets of pixels. The F-test is a statistical test which is used to test if the variances of two populations are equal or not.

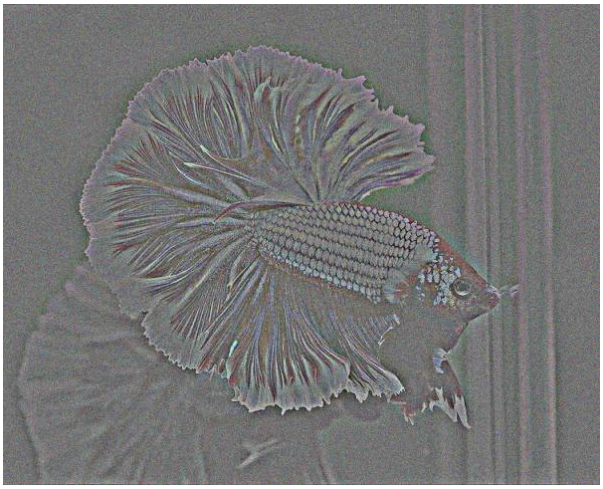
We use the following hypotheses for our F-test:

$$H_0 : \sigma_1^2 = \sigma_2^2$$

$$H_1 : \sigma_1^2 \neq \sigma_2^2$$



(a) $\sigma = 1$



(b) $\sigma = 5$

(c) $\sigma = 10$

Figure 3.4: Residuals of different blurred images from figure 3.3

The F-value is calculated using the following formula:

$$F = \frac{s_1^2}{s_2^2}$$

where s_1^2 and s_2^2 are the sample variances of two images. From the F-value, we can calculate the p-value for the hypothesis test. To simplify the calculation, the function *var.test* in R is used to find both F-value and p-value. To perform the F-test, the assumption about the normal distribution of the image residual and the independent between the image residual and the picture itself is necessary.

3.3 Pitman's Test: Comparing Variances of Correlated Samples

The test applies for hypothesis claims to test if two correlated samples have common variance or not. The null hypothesis for Pitman's test is that both samples come from populations with the same variance. The t-value of the test will be computed as following

$$t = \frac{(F-1)\sqrt{n-2}}{2\sqrt{F(1-r)^2}}$$

The parameter F is the ratio between the variance of both samples while the variable r is the correlation of those samples [12]. The correlation of two samples x and y can be computed using the Pearson correlation coefficient as below.

$$r = \frac{\sum_{i=1}^{n}(x_i-\bar{x})(y_i-\bar{y})}{\sqrt{\sum_{i=1}^{n}(x_i-\bar{x})^2(y_i-\bar{y})^2}}$$

Then, we can compare the t-value to the significant value to make the conclusion about the hypothesis claim [12].

To apply Pitman's Test, two samples need to be correlated; if they are not, the test should not be used. We can apply this test to compare if the noise levels of both images of the same object which come from two resource. Since the objects on both images are the same, two image are correlated.

CHAPTER IV: NUMERICAL RESULTS

4.1 Image data creation

To create the image data for the test, an arbitrary image was picked as the original image. Then, the noise was added to the image using different models. There are two main models that generated the noisy image, an independent noise model and a dependent noise model. The detail of these two models are described in the subsection below.

4.1.1 Independent noise models

For independent noise, the pixels position and their intensity still do not affect to the noise level [5]. Each pixel is modified a random amount using a probability distribution with constant variance. The most popular independent noise model is Gaussian noise, described as:

$$y_{i,j} = x_{i,j} + b \quad \text{with} \quad b \sim N(0, \sigma^2)$$

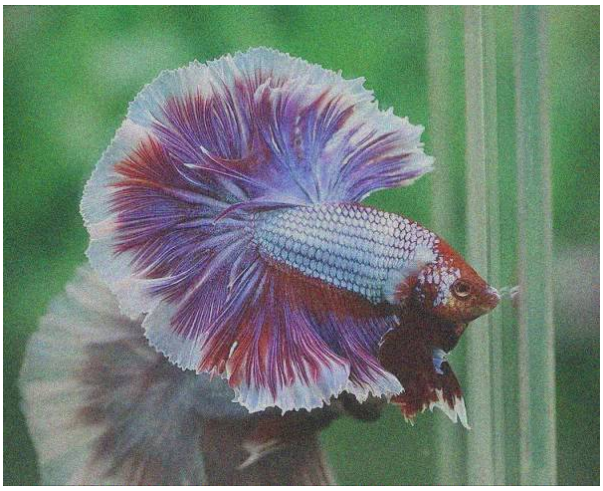
where y is the generated noisy image, x is the original image and b is the additive Gaussian noise[6]. In which, the noise element is normal distributed with zero mean and a specific standard deviation. The range of numerical values for a pixel is [0, 255]; therefore, if the added noise takes a pixel value out of this range, it is set to the relevant boundary value. An example of this type of noise model is shown in figure 4.8a. Two standard deviations were chosen which is 0.3 and 0.6, and fifty images were generated for comparison using each standard deviation.

4.1.2 Dependent noise models

The second noise model deals with dependent noise. The noise level of this model depends on the intensity of the image [8]. The dependent noise level model used was

$$y_{i,j} = x_{i,j} + x_{i,j}^{\mu} * u + w$$

(a) Original Image



(b) Independent Noise Image with $\sigma = 0.3$ and $\mu = 0$

(c) Dependent Noise Image with $\sigma = 0.3$, $\mu = 0$, $\sigma_u = 5$, $\sigma_w = 1.5$

Figure 4.5: Original image and noisy images

where y is the noisy image, x is the original image while x_i is the sub-image, μ is the exponential parameter, and u and w are zero mean random variables with variance σ_u^2 and σ_w^2 [8]. Figure 4.5c shows the dependent noise image of the original from figure 4.7a.

4.1.3 Multivariate noise models

The noise of two different images are sometime related when they are affected by the same cause of noise. Therefore, we need to create samples of images that have correlated noise. We assume that the noise is random; as a result, normal distribution is used one more time. However, the regular normal distribution just has one variable. Then, we can use bivariate normal distribution in this case to make up of two independent random variable.

We need to generate two random noise effects which are correlated. The first step is generating the correlation matrix as following.

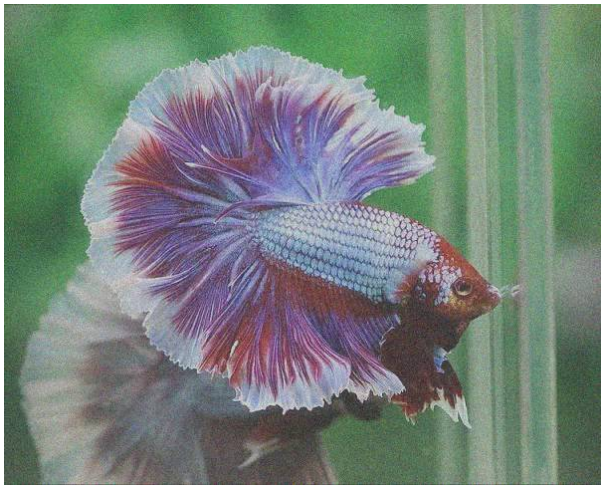
$$\Sigma = \begin{bmatrix} 1 & \rho \\ \rho & 1 \end{bmatrix}$$

where ρ is the correlation between the noise of the two images. We can set this parameter in the range from 0 to 1 depending how much correlation we want between the images. If the coefficient is close to 0, the images are largely uncorrelated, while if it is near 1, the images are highly correlated. After the correlation matrix is set, we can generate the noise to add into the original image. The two generated noise levels are added directly to the root image as independent noise. Figure 4.6 below has examples of two correlated images with coefficient of 0.7.

4.2 Estimation of image noise and sub-sampling method

4.2.1 Noise level estimation using WTP

Since the scale of the noise level is unknown, we can create some threshold for calculation. The threshold noise level column is the noise level that is determined at the beginning. Different thresholds are chosen to compare the correctness of the algorithm. The noise estimation for three channels are calculated and the average of them is computed.



(a) First correlated noise image with $\sigma = 0.3$

(b) Second correlated noise image with $\sigma = 0.3$

(c) First correlated noise image with $\sigma = 0.6$

(d) Second correlated noise image with $\sigma = 0.6$

Figure 4.6: Correlated noise image with correlation coefficient of $\rho = 0.7$

Threshold Noise Level	WTP Estimation
5	5.3192
10	10.3698
15	15.2716
20	20.1991

4.2.2 Sub-sampling Method

From a noisy image, multiple sub-images are picked randomly. The size of each sub-image is fixed in this paper at 200×200 pixels. After a sub-image is created, we can find the variance of that subsample. Then, the process is repeated multiple times to collect multiple estimates of the variance of the image. We can also calculate the confidence interval of the variance depending on the result of estimated range. The table below shows the range of the variance on each color channel for 500 sub-images taken from a noise image.

Color channel	Average	95% of confidence Interval
Red	1.6653	(1.5899; 1.7407)
Green	1.5984	(1.5213; 1.6754)
Blue	1.5945	(1.5172; 1.6719)

The range of the variance is also not too large. This means that we can compare the range of the image variance to determine how different they are. We will compare the variance of two images and the result is shown below.

4.3 Comparison of the noise level

4.3.1 F-test performance

To determine if the hypothesis claim of equal variance is true or not, we can use F-test which is based on F-distribution. After we apply the WTP to find the variance of two images, we will apply the F-test to the ratio of these two values to determine if the underlying images have the same variance or not.

To form the traditional F-test, there are a few more parameters that we have to determine for the F-test: the significance level α, the degrees of freedom of both image and the F table critical value. The F critical value is computed by using the formula described on the previous section. We have to pay attention that the F critical value has to be greater than one because we need to pick the larger variance for numerator. If calculated F value is less than 1, we need to flip the formula by put one over the result. For the alpha level, we determine that 95 percent of confidence interval is good enough for the test and the alpha value is picked at 0.05 for two-tail test. The degrees of freedom are determined as the images size in pixels. We can determine the F table value now by looking up the F-table of $\alpha = 0.05$ and use the degrees of freedom to find the corresponding value. The degree freedom of the variance placed in the numerator when finding the F critical value is used to find the row of the F-table. When the F critical value and the F-table value are ready, we will compare these two results to make a conclusion about the null hypothesis. The claim is rejected if the F critical value is larger than the F-table value. One of the results is show below when comparing two image variances. The result is calculated by using *pf* function in R.

Color channel	F-value	p-value	Conclusion
Red	24.4151	0	Reject H_0
Green	19.90258	0	Reject H_0
Blue	40.2167	0	Reject H_0

When the degrees of freedom are too large for the table (here, 483×600), the F-table value is 1.0. The p-value for each color channel is calculated as the table above. The result show that all the p-values are all zeros when the degree of freedom is too large. This leads the p-values less than the significant level. This means the null hypothesis of each channel is rejected. This means the variance of two images is not equal. Since two images are generated with the same independent noise level, this conclusion tells us that the F-test on the image variances seems to be incorrect. The reason might come from the residual image which is the difference between the noise image and its smooth image. These estimated residuals are not independent and they are different to true noise random variable.

The assumption for the test is violated; therefore, we need a better method to compare the image noise.

4.3.2 Pittman T Test

As an alternative to the F-test, we consider comparing images using Pitman's t-test.

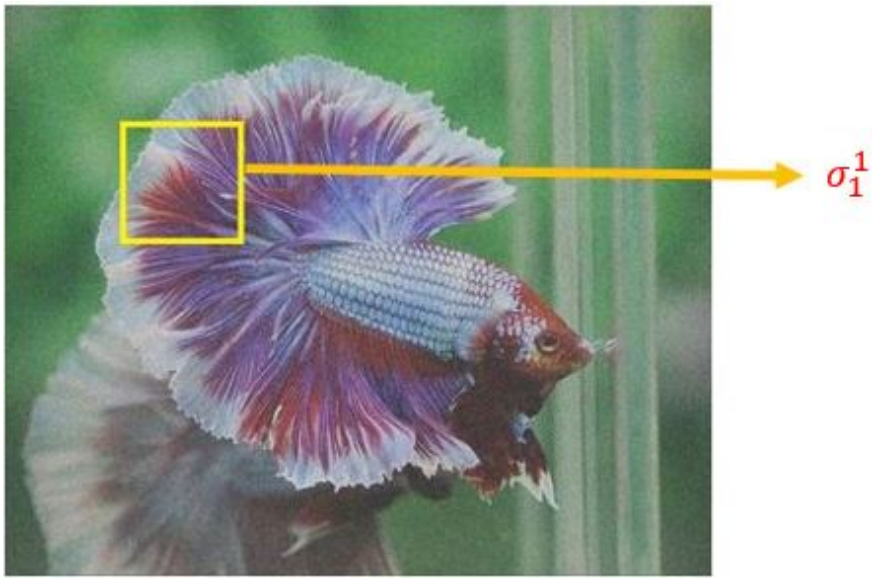
For this type of test, we need to determine the parameters of the test which are the F value and the correlation r [12]. The F value is the ratio of the variances of two images which are obtains through Section 2.1. Then, we need to determine the correlation between two images. To calculate the correlation of two image noise, we used formula on the residual of the image. Then, the p-value is calculated using the pf function in R using the degree of freedom equal to image size. The result of the Pittman T Test is shown in the table below.

Color channel	F ratio	t-value	p-value	Conclusion
Red	24.4151	$8.6226e+03$	$< .00001$	Reject H_0
Green	19.90258	$8.5545e+03$	$< .00001$	Reject H_0
Blue	40.2167	$1.2117e+04$	$< .00001$	Reject H_0

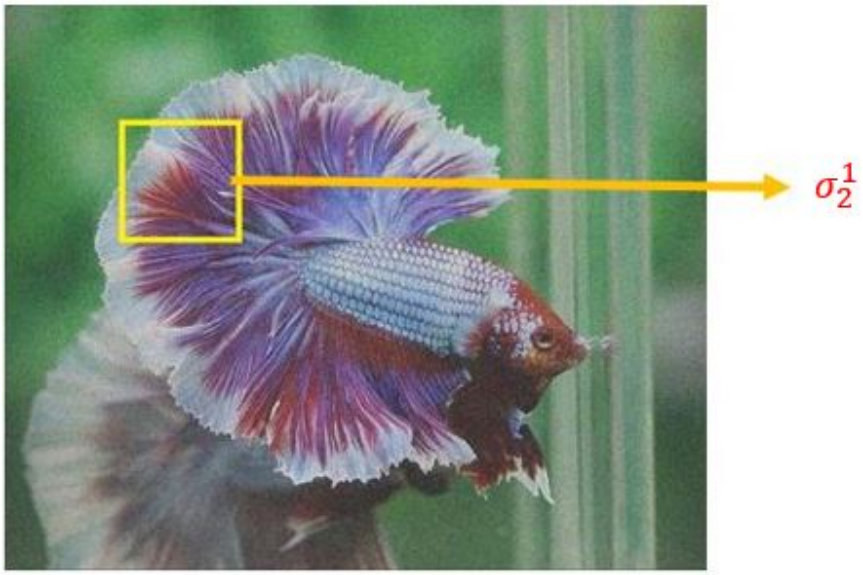
Depending on the result of the t-test, we reject the null hypothesis which says that two images have common variance.

4.3.3 Paired sub-sampling method for bivariate normal noise

Since we know that two images are correlated, we would like to test if the noise levels are the same or not. We did not perform F-test on this difference vector when it does not work well to compare the image noise level on previous chapter. To perform the test, we need to generate the noise data of each sub-image. The sub-image is pick randomly as the description in section 3.4. However, we will perform sub-sampling for two noisy images. The random sub-samples are chosen to be the same for each image as in Figure 4.7 before we calculate the noise levels σ_1 and σ_2. The size of the sub-image is still the same, 200×200 pixels. We also still collect 500 samples for each image to calculate the



(a) First correlated noise image



(b) Second correlated noise image

Figure 4.7: Paired Subsampling test

noise level. The output will be a matrix 500×2 which contains the noise level of both images for the 500 sub-samples.

Next step, we will perform paired subsampling by finding the difference between noise level of two images by subtracting the first column vector by second column vector. From two vectors of

calculated variance in Figure 4.7, difference vector can be determined by subtracting these vectors as following

$$\hat{\sigma} = \sigma_1 - \sigma_2$$

where $\hat{\sigma}$ is the difference vector. By using this vector, we can find the distribution of the difference between two variance vector and construct the confidence interval for the variance.

The procedure is as following:

Paired Subsampling Procedure:

1. Pick a sub-sample randomly.

2. Estimate the variance using WTP.

3. Repeat step 1 and 2 for 500 times.

4. Calculate the difference vector and find the distribution of 500 results to approximate the sampling distribution.

5. Construct a confidence interval of 95%.

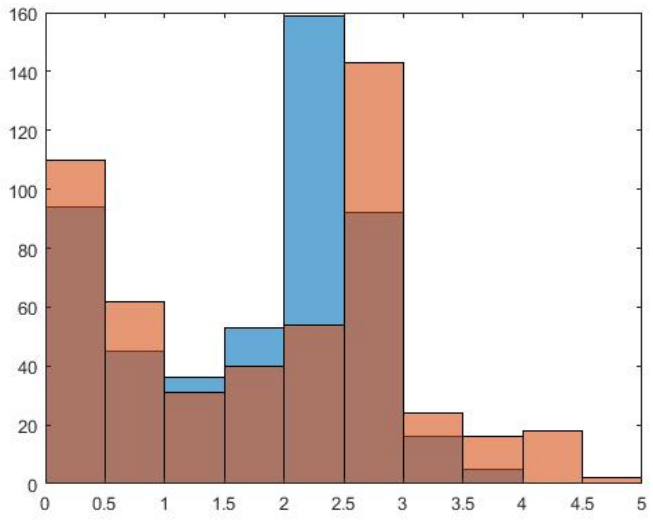
The confidence interval is constructed from the difference vector by the percentile method and the 95% confidence interval is the range of points that cover the middle 95% of the sampling distribution.

The vector of difference will be used to calculate the p-value for the hypothesis test. The percentage of the sampling distribution below zero is determined. The formula is described below:

$$p - value = \begin{cases} 2 \times (1 - P(x < 0)) & \text{if P(x<0) >0.5} \\ 2 \times (P(x < 0)) & \text{if P(x<0) <0.5} \end{cases}$$

The null hypothesis is that the subsamples share the same level of noise. The result of the t-test will lead us to reject the null hypothesis or not. A histogram of the estimated noise levels in the pairs is shown in Figure 4.8.

Based on the histogram, we can conclude qualitatively that the two images have the same noise level since the histograms have substantial overlap with each other when correlation is 0.7. The red



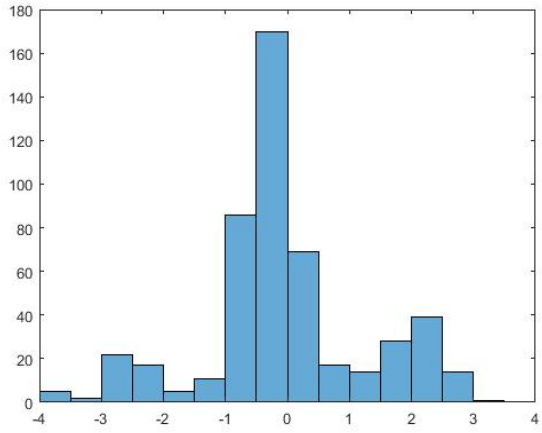
(a) $\rho = 0.7, \sigma = 0.3$

Figure 4.8: Histogram of the calculated variance on both bivariate noise images

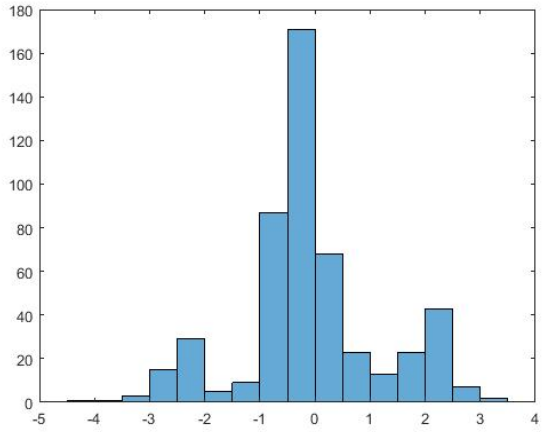
histogram is the noise level of one image and the blue histogram belongs to the other correlated noise image. To make the comparison precise, we will find the difference between two vectors of subsample variances and calculate a t-value. The histogram of the distributions of differences is shown in figure 4.9.

The result show that the mean of the differences is (0.4990, 0.4813, 0.4892) and the standard deviation is (1.1918, 1.1985, 1.2098) for three color channels. All confidence intervals contain zero and the approximate p-values are greater than 0.05. As a result, we appropriately fail to reject the null hypothesis of equal variance.

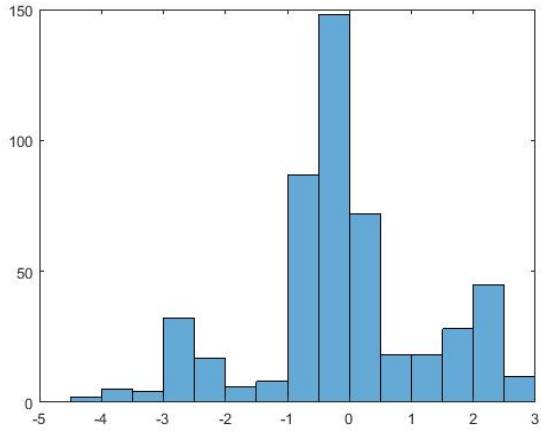
`Color channel`	Mean	p-value	Confidence Interval
`Red`	`-0.0531`	0.792	(-2.7086,2.5358)
`Green`	`-0.0787`	0.716	(-2.6716,2.4436)
`Blue`	`-0.1235`	0.764	(-2.9647,2.4559)



(a) Red Channel



(b) Blue Channel



(c) Green Channel

Figure 4.9: Histogram of difference vector

CHAPTER V: SUMMARY AND CONCLUSIONS

The image noise is one of important attributes of the image quality. In the project, we reviewed some models for the image noise and the WTP method for the estimation of the image noise level. The WTP estimates are not simple which are based on PCA and weak texture patches. Therefore, it is difficult to derive their sampling distributions which are required for statistical inference. We proposed a computational method to obtain the sample distributions by generalizing one dimensional subsampling to a two-dimensional version. The confidence intervals of the noise levels can be constructed. For numerical results, some testing images were generated based on one fish image using different noise models such as white Gaussian noise, signal dependent noise and bivariate noise models. Traditionally, F-tests and Pitman tests were used to compare the variances. We applied all these tests as well as the subsampling methods to compare the noise levels of these testing images. We found that F-tests and Pittman test do not work well. Subsampling methods are able to provide some reasonable results and can be used for testing other type of dependent data. For future work, the use of parallel computing is possible to improve the processing speed when comparing variance of the images.

REFERENCES

[1] "Concept of pixel." https://www.tutorialspoint.com/dip/concept$_of_pixel.htm$. *Accessed* : 2019 – 03 – 08.

[2] S. Nair, S. Ha, and W. Xu, "Data analysis on multivariate image set," in *2018 New York Scientific Data Summit (NYSDS)*, pp. 1–3, Aug 2018.

[3] "Getting started with imager." https://cran.r-project.org/web/packages/imager/vignettes/getting-started.html. Accessed: 2019-01-25.

[4] D. Temel, M. Prabhushankar, and G. AlRegib, "Unique: Unsupervised image quality estimation," *IEEE Signal Processing Letters*, vol. 23, pp. 1414–1418, Oct 2016.

[5] G. Chen, F. Zhu, and P. A. Heng, "An efficient statistical method for image noise level estimation," in *2015 IEEE International Conference on Computer Vision (ICCV)*, pp. 477–485, Dec 2015.

[6] C. Liu, W. T. Freeman, R. Szeliski, and S. Bing Kang, "Noise estimation from a single image," in *2006 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'06)*, vol. 1, pp. 901–908, June 2006.

[7] X. Liu, M. Tanaka, and M. Okutomi, "Noise level estimation using weak textured patches of a single noisy image," in *2012 19th IEEE International Conference on Image Processing*, pp. 665–668, Sep. 2012.

[8] X. Liu, M. Tanaka, and M. Okutomi, "Signal dependent noise removal from a single image," in *2014 IEEE International Conference on Image Processing (ICIP)*, pp. 2679–2683, 2014.

[9] X. Liu, M. Tanaka, and M. Okutomi, "Single-image noise level estimation for blind denoising," *IEEE Transactions on Image Processing*, vol. 22, pp. 5226–5237, Dec 2013.

[10] X. Zhu and P. Milanfar, "Automatic parameter selection for denoising algorithms using a no-reference measure of image content," *IEEE Transactions on Image Processing*, vol. 19, pp. 3116–3132, Dec 2010.

[11] R. Sadek, "Svd based image processing applications: State of the art, contributions and research challenges," Nov 2012.

[12] G. S. Mudholkar, G. E. Wilding, and W. L. Mietlowski, "Robustness properties of the pitmanmorgan test," *Communications in Statistics - Theory and Methods*, vol. 32, no. 9, pp. 1801–1816, 2003.

[13] Z.-J. Ding, Y. Zhang, A.-Q. Yang, and Dai-Li, "Image matching of gaussian blurred image based on sift algorithm," in *2012 International Conference on Wavelet Active Media Technology and Information Processing (ICWAMTIP)*, pp. 121–124, Dec 2012.

APPENDIX A: CODE

6.1 Matlab Code

6.1.1 Noise Estimation

```
img = double(imread('RN03_1.jpg'));
mskflg = 0; % if you want to produce the mask, put one for mskflg
level = [5,10,20,40];
for i=1:size(level,2);
 noise = img;
 tic;
 [nlevel th] = NoiseLevel(noise);
 t=toc;
 fprintf('True: %5.2f  R:%5.2f G:%5.2f B:%5.2f\n', level(i), nlevel(1),
    nlevel(2), nlevel(3) );
 fprintf('Calculation time: %5.2f [sec]\n\n', t );

 if( mskflg )
   msk = WeakTextureMask( noise, th );
   imwrite(uint8(msk*255), sprintf('msk%02d.png', level(i)));
 end
end
```

6.1.2 Variance calculation

```
    function [nlevel th num] = NoiseLevel(img,patchsize,decim,conf,itr)
if( ~exist('itr', 'var') )
    itr = 3;
```

```
end
if( ~exist('conf', 'var') )
    conf = 1-1E-6;
end
if( ~exist('decim', 'var') )
    decim = 0;
end
if( ~exist('patchsize', 'var') )
    patchsize = 7;
end
kh = [-1/2,0,1/2];
imgh = imfilter(img,kh,'replicate');
imgh = imgh(:,2:size(imgh,2)-1,:);
imgh = imgh .* imgh;
kv = kh';
imgv = imfilter(img,kv,'replicate');
imgv = imgv(2:size(imgv,1)-1,:,:);
imgv = imgv .* imgv;
Dh = my_convmtx2(kh,patchsize,patchsize);
Dv = my_convmtx2(kv,patchsize,patchsize);
DD = Dh'*Dh+Dv'*Dv;
r = rank(DD);
Dtr = trace(DD);
tau0 = gaminv(conf,double(r)/2, 2.0 * Dtr / double(r));
%{
eg = eig(DD);
tau0 = gaminv(conf,double(r)/2, 2.0 * eg(patchsize*patchsize));
%}
```

```
for cha=1:size(img,3)
X = im2col(img(:,:,cha),[patchsize patchsize]);
Xh = im2col(imgh(:,:,cha),[patchsize patchsize-2]);
Xv = im2col(imgv(:,:,cha),[patchsize-2 patchsize]);

Xtr = sum(vertcat(Xh,Xv));
if( decim > 0 )
    XtrX = vertcat(Xtr,X);
    XtrX = sortrows(XtrX')';
    p = floor(size(XtrX,2)/(decim+1));
    p = [1:p] * (decim+1);
    Xtr = XtrX(1,p);
    X = XtrX(2:size(XtrX,1),p);
end
%%%%% noise level estimation %%%%%
    tau = Inf;
    if( size(X,2) < size(X,1) )
        sig2 = 0;
    else
        cov = X*X'/(size(X,2)-1);
        d = eig(cov);
        sig2 = d(1);
    end

for i=2:itr
%%%%% weak texture selectioin %%%%%
    tau = sig2 * tau0;
    p = (Xtr<tau);
```

```
    Xtr = Xtr(:,p);
    X = X(:,p);

    %%%%% noise level estimation %%%%%
        if( size(X,2) < size(X,1) )
            break;
        end
    cov = X*X'/(size(X,2)-1);
    d = eig(cov);
    sig2 = d(1);
end
nlevel(cha) = sqrt(sig2);
th(cha) = tau;
num(cha) = size(X,2);
end
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%%
function T = my_convmtx2(H, m, n)
s = size(H);
T = zeros((m-s(1)+1) * (n-s(2)+1), m*n);
k = 1;
for i=1:(m-s(1)+1)
 for j=1:(n-s(2)+1)

  for p=1:s(1)
   T(k,(i-1+p-1)*n+(j-1)+1:(i-1+p-1)*n+(j-1)+1+s(2)-1) = H(p,:);
  end
```

```
  k = k + 1;
 end
end
```

6.1.3 Confidence Interval Calculation

```
    %Find confidence interval of variance from sub-images
clc
clear

Noise = []
%Input image
original_image = imread('RN03_1.jpg');

[width, length, deep] = size(original_image);
sub_size = 50;
loop = 500


for i = 1:loop
     x=round(1+rand()*(width));
     y=round(1+rand()*(length));

     if(x + sub_size > width)
         x = width - sub_size;
         display("Replace x");
     end

     if(y + sub_size > length)
```

```
        y = length - sub_size;
        display("Replace y");
    end
    %Generate sub image
    sub_image = original_image(x:x+sub_size-1,y:y+sub_size-1,:);
    %Calculate Noise Level
    [nlevel th] = NoiseLevel(double(sub_image));

    Noise(i,:) = nlevel;
end

SEM = std(Noise)/sqrt(loop);               % Standard Error
ts = tinv([0.025  0.975],loop-1);      % T-Score
CI_low = mean(Noise) + ts(1)*SEM;                    % Confidence Intervals
CI_high = mean(Noise) + ts(2)*SEM;
```

6.1.4 Pair subsampling test

```
clc
clear

Noise1 = []
Noise2 = []
%Input image
noise_image_1 = imread('Mul_fish06_1.jpg');
noise_image_2 = imread('Mul_fish06_2.jpg');

[width, length, deep] = size(noise_image_1);
```

```
sub_size = 100;

loop = 500


for i = 1:loop
     x=round(1+rand()*(width));
     y=round(1+rand()*(length));

     if(x + sub_size > width)
         x = width - sub_size;
         display("Replace x");
     end

     if(y + sub_size > length)
         y = length - sub_size;
         display("Replace y");
     end
     %Generate sub images
     sub_image_1 = noise_image_1(x:x+sub_size-1,y:y+sub_size-1,:);
     sub_image_2 = noise_image_2(x:x+sub_size-1,y:y+sub_size-1,:);

     %Calculate Noise Level
     [nlevel1 th1] = NoiseLevel(double(sub_image_1));
     [nlevel2 th2] = NoiseLevel(double(sub_image_2));

     Noise1(i,:) = nlevel1;
     Noise2(i,:) = nlevel2;
```

```
end


Dif = Noise1-Noise2;

avr = mean(Dif)

stand = std(Dif)

%Performing paired $t$-test

A=sort(Noise1(:,1))

B=sort(Noise2(:,1))

ttest(A(12:487),B(12:487))

ttest(Dif)
```

6.2 R Code

6.2.1 Bivariate Image Generation

```
rm(list = ls())

library(imager)

library(purrr)

library(ggplot2)

library(dplyr)

library(RandomFields)

library(MASS)

library(mvtnorm)




#Root Image

#Choosing the file location
```

```
file.location = file.choose()
#Input the image
fish<-load.image(file.location)

rou = 0.3
sig = matrix(c(1,rou,rou,1), ncol = 2, nrow = 2)

Mul_r = rmvnorm(600*483,sigma = sig)
Mul_g = rmvnorm(600*483,sigma = sig)
Mul_b = rmvnorm(600*483,sigma = sig)

const = 0.6

newfish1 = fish
newfish1[,,,1] = newfish1[,,,1]+Mul_r[,1]*const^2
newfish1[,,,2] = newfish1[,,,2]+Mul_g[,1]*const^2
newfish1[,,,3] = newfish1[,,,3]+Mul_b[,1]*const^2

newfish2 = fish
newfish2[,,,1] = newfish2[,,,1]+Mul_r[,2]*const^2
newfish2[,,,2] = newfish2[,,,2]+Mul_g[,2]*const^2
newfish2[,,,3] = newfish2[,,,3]+Mul_b[,2]*const^2



plot(newfish1)
save.image(newfish1,"Mul_fish06_21.jpg")

plot(newfish2)
```

```
save.image(newfish2,"Mul_fish06_22.jpg")
```

6.2.2 White Gaussian Noise Creation

```
    #Loading the library
rm(list = ls())
library(imager)
library(purrr)
library(ggplot2)
library(dplyr)
library(RandomFields)
library(MASS)
library(mvtnorm)
library(boot)

#Choosing the file location
file.location = file.choose()

#Input the image
fish<-load.image(file.location)

#Find the mean value of each channel on the image
mean<-c(mean(fish[,,,1]),mean(fish[,,,2]),mean(fish[,,,3]))

#Standard deviation vector
sd <- c(sd(fish[,,,1]),sd(fish[,,,2]),sd(fish[,,,3]))

#find the random noise for each channel, set mean is 0, control std
```

```
#random filed of the same type (related, correlation)
#difference , mean square(diff^2/total)



for (i in 1:50) {

  rnoise = rnorm(600*483, mean(fish[,,,1]), 0.6^2)
  gnoise = rnorm(600*483, mean(fish[,,,2]), 0.6^2)
  bnoise = rnorm(600*483, mean(fish[,,,3]), 0.6^2)

#Generate new fish and adding the noise into
newfish = fish
newfish[,,,1] = newfish[,,,1]+rnoise
newfish[,,,2] = newfish[,,,2]+gnoise
newfish[,,,3] = newfish[,,,3]+bnoise

#Save the fish
#plot(newfish)
#dev.copy(jpeg,filename="fish3.jpg")

#Blur the fish
#blurfish <- isoblur(fish,5)

#Create the file name
#file_name = paste0("RNFish06x2_",toString(i),".jpg")

#Save the noise added:
```

```
#save.image(newfish,file_name)
save.image('RN')
save.image(newfish,'RN06_2.jpg')
}
#Histogram plot
#bdf <- as.data.frame(fish)
#bdf <- mutate(bdf,channel=factor(cc,labels=c('R','G','B')))
#ggplot(bdf,aes(value,col=channel))+geom_histogram(bins=30)+facet_wrap(~ channel)



#imnoise(dim=dim(fish))
#gauss_fish = fish+imnoise(dim=dim(fish), sd = 0.1)
#save.image(gauss_fish,"fish44.jpg")

dmv1 = dmvnorm(newfish[,,,1])
dmv2 = dmvnorm(newfish[,,,2])
dmv3 = dmvnorm(newfish[,,,3])

newfish[,,,1] = newfish[,,,1]+dmv1
newfish[,,,2] = newfish[,,,2]+dmv2
newfish[,,,3] = newfish[,,,3]+dmv3

plot(newfish)
plot(fish)

rsq <- function(formula, data, indices) {
  d <- data[indices,] # allows boot to select sample
  fit <- lm(formula, data=data=newfish[,,,1])
```

```
  return(summary(fit)$r.square)
}
results <- boot(data=newfish[,,,1], statistic = rsq, R=500)
```