Spring 5-1-2021

Efficient Denoising Of High Resolution Color Digital Images Utilizing Krylov Subspace Spectral Methods

Eva Lynn Greenman
University of Southern Mississippi

Spring 5-5-2021

Efficient Denoising Of High Resolution Color Digital Images Utilizing Krylov Subspace Spectral Methods

Eva Lynn Greenman

EFFICIENT DENOISING OF HIGH RESOLUTION COLOR DIGITAL IMAGES UTILIZING KRYLOV SUBSPACE SPECTRAL METHODS

by

Eva Lynn Greenman

A Dissertation
Submitted to the Graduate School,
the College of Arts and Sciences
and the School of Mathematics and Natural Sciences
of The University of Southern Mississippi
in Partial Fulfillment of the Requirements
for the Degree of Doctor of Philosophy

Approved by:

Dr. James V. Lambers, Committee Chair
Dr. Jiu Ding
Dr. Haiyan Tian
Dr. Huiqing Zhu

May 2021

ABSTRACT

The solution to a parabolic nonlinear diffusion equation using a Krylov Subspace Spectral method is applied to high resolution color digital images with parallel processing for efficient denoising. The evolution of digital image technology, processing power, and numerical methods must evolve to increase efficiency in order to meet current usage requirements. Much work has been done to perfect the edge detector in Perona-Malik equation variants, while minimizing the effects of artifacts. It is demonstrated that this implementation of a regularized partial differential equation model controls backward diffusion, achieves strong denoising, and minimizes blurring and other ancillary effects. By adaptively tuning edge detector parameters so as to not require human interaction, we propose to automatically adapt the parameters to specific images. It is anticipated that with KSS methods, in conjunction with efficient block processing, we will set new standards for efficiency and automation.

ACKNOWLEDGMENTS

I thank my daughters Sarah, Christina, Sophia and Julia and for their encouragement, support and patience as I devoted my time and attention towards my studies. I thank my friends near and far for their positive support as I traveled this long and winding road. I thank my fellow graduate students whose counsel and humor motivated, inspired and assured me that I was not alone. I am forever indebted to my advisor Dr. James Lambers and to my doctoral committee members Dr. Jiu Ding, Dr. Haiyan Tian and Dr. Huiqing Zhu for their patience, guidance and inspiration.

TABLE OF CONTENTS

LIST OF ILLUSTRATIONS

Figure

LIST OF TABLES

Table

LIST OF ABBREVIATIONS

PDE	-	Partial Differential Equation
AD	-	Anisotropic Diffusion
PME	-	Perono-Malik Equation
RGB	-	Red, Green and Blue Colors
PSNR	-	Peak Signal to Noise Ratio
SSIM	-	Structural Similarity Measure
SSIM*	-	Scaled Structural Similarity Measure
MSE*	-	Relative Mean Square Error
MSE	-	Mean Square Error
TQ	-	Total Quality Measure
pTQ	-	Percent change of Total Quality Measure
Blk	-	Image Sub block such as: R11
MTHs	-	Soft Thresholding Denoising
MTHh	-	Hard Thresholding Denoising
MMV	-	Moving Median Denoising
MW	-	Weiner Filter Denoising
MS	-	Convolutional Smoothing Denoising
IDF	-	Diffusion Filter Denoising
KSS	-	KSSFast With Parameter Optimization

There are two more specific abbreviation tables in Chapter 4, on page 50 before the figures and on and page 139 before the tables.

NOTATION AND GLOSSARY

General Usage and Terminology

The notation used in this text represents fairly standard mathematical and computational usage. In many cases these fields tend to use different preferred notation to indicate the same concept, and these have been reconciled to the extent possible, given the interdisciplinary nature of the material. In particular, the notation for partial derivatives varies extensively, and the notation used is chosen for stylistic convenience based on the application. While it would be convenient to utilize a standard nomenclature for this important symbol, the many alternatives currently in the published literature will continue to be utilized.

The blackboard fonts are used to denote standard sets of numbers: $\mathbb{R}$ for the field of real numbers, $\mathbb{C}$ for the complex field, $\mathbb{Z}$ for the integers, and $\mathbb{Q}$ for the rationals. The capital letters, $A, B, \cdots$ are used to denote matrices, including capital greek letters, e.g., Λ for a diagnonal matrix. Functions which are denoted in boldface type typically represent vector valued functions, and real valued functions usually are set in lower case roman or greek letters. Caligraphic letters, e.g., $\mathcal{V}$, are used to denote spaces such as $\mathcal{V}$ denoting a vector space, $\mathcal{H}$ denoting a Hilbert space, or $\mathcal{F}$ denoting a general function space. Lower case letters such as i, j, k, l, m, n and sometimes p and d are used to denote indices.

Vectors are typset in bold lowercase , e.g., $\mathbf{u}$, and matrices are typeset in square brackets, e.g., $[\cdot]$. In general the norms are typeset using double pairs of lines, e.g., $||\cdot||$. The inner product operator is typeset using left and right angles, e.g., $\langle \cdot, \cdot \rangle$.

Chapter 1

INTRODUCTION

1.1 Motivation

In 1957 the first scanner created the first digital image and nearly twenty years later the first digital camera captured the first digital photograph. Spurred by these two milestones, the quest to seize upon the utility of digital images motivated mathematicians and computer scientists to find methods to optimize image quality. Constant advances in computer processing and storage capabilities compel image processing methods to keep pace and ensure the area of research remains relevant. In 2020 there are estimated to be 1.4×10^{12} digital images [1]. This massive quantity of images necessitates efficient methods to enable their timely use.

A primary step of digital image processing is noise removal. Every digital image is imperfect and contains noise which diminishes the utility of poor quality images. This problem is both unavoidable and impossible to solve completely [19]. Nonetheless, investments of time and expense to capture images warrants solving the problem. An acceptable balance must be found to maximize noise removal and retention of image features. Ideally this is done without introducing artifacts, which are any new features added by processing that are not present in the original image. Many satellite images in their raw form contain so much noise that useful features are not detectable until after considerable processing.

Noise results from image sensor technology and the environment in which the images are captured. Evolution has enabled human vision to adapt, allowing the useful interpretation of even poor quality images [48]. Although the subjective evaluation of an image by a human observer is the best method to assess images quality, this not always practical or efficient [26]. Automated quality measures and denoising methods are necessary for algorithms to utilize the vast store of digital images in need of processing.

There are many methods available to denoise images, including filtering, averaging, transforms and partial differential equation (PDE) methods. This research utilizes a regularized anisotropic diffusion equation, a modified Perona-Malik equation, and Krylov Subspace Spectral methods to enhance efficiency and effectiveness. In addition, iterative adaptive parameter tuning, an improved image quality measure and a parallel blocking

scheme allow for efficient and localized improvement customized to any image. Comparison of the proposed image processing will be done with representative "state of the art" denoising methods to show the effectiveness. It is expected that results will show improved performance in processing time, image quality measures, reduction in method artifacts due to a more localized application of adaptive parameter settings, and block processing.

The remainder of Chapter 1 will contain background information on digital images, noise, image quality measurement, and "state of the art" denoising techniques. Chapter 2 contains a literature review of image desnoising methods in two parts. The first is an evolutionary view of PDE-based aniostropic diffusion methods. The second is an overview of PDE solution methods, ending with Krylov Subspace Spectral (KSS) methods. Chapter 3 outlines the proposed algorithm. Chapter 4 contains results of the proposed method as well as results for benchmark methods. Chapter 5 contains a summary of results with conclusions and future work. Appendix A contains a description of the noise quantification methods used to formulate a total quality measure used in the adaptive parameter algorithm. Appendix B contains a brief description of comparative methods against which the proposed method is compared.

1.2 Background

Before providing a detailed explanation of the method, it will be helpful to introduce some background. Pertinent details of digital images, noise, measures of image quality, and a summary of state of the art denoising methods will be provided. Digital images are created as spatially arranged digital light sensors capture light intensities, allowing the real image to be stored as pixels. A real image is actually a continuous spectrum of light reflecting off the surface of an object. Digital images are discretizations of continuous images. A pixel (**pic**-ture and **el**-ement) is the smallest unit of that discretization. The resolution of a digital image describes the number of pixels in horizontal and vertical dimensions. Another reference to resolution can be a single number listing the storage need for an image, such as 5 Mega Pixels.

Image pixels are spatially arranged by rows and columns that correlate to the location of real image features. Each pixel value is a single number recording the magnitude of color intensity, stored in one of three mono-color-referenced matrices. One digital image can be represented by three two-dimensional matrices. Color images have pixel intensities ranging from 0 for the lack of any color to 255 for maximum of red, green or blue. A black and white image is represented by a single matrix with pixel intensity values of 0 or 1. A grayscale image is a numerical conversion of the three color image format that is scaled and

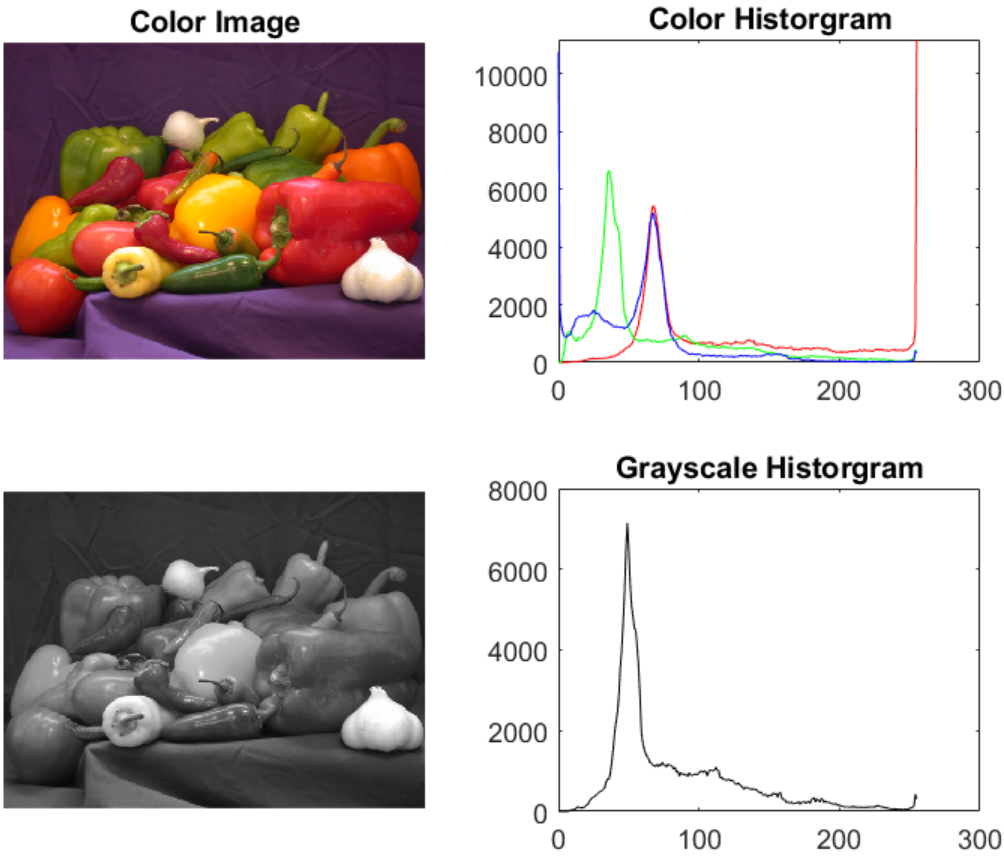

Figure 1.1: Pepper RGB Image (top left) and Associated Pixel Intensity Histogram (top right) and Grayscale Image (bottom left) and Associated Pixel Intensity Historgram

color referenced into a single two dimensional matrix. A grayscale image matrix has pixel intensities that range from 0 for black to 255 for white, and enable a full spectrum of gray values in between.

In this research all images will be in RGB color format, for which each color matrix will be processed separately. In a sense, each color matrix is a unique image representing a particular color range. When all three matrices are combined, the full three-color effect is realized. Processing each color matrix separately is more exact because noise can affect different color matrices in different ways [37]. Specifically, the blue image matrix for a digital image can be particularly susceptible to noise due to amplification in blue light sensors. Experimentally created noisy images won't vary necessarily between the color matrices, but real noisy images, towards which this research is aimed, we expect will benefit from this individualized treatment of color matrices. Typically denoising research has concentrated on

denoising Grayscale images, rather than denoising each color matrix independently, to avoid the effects of blurring which can result in color artifacts that introduce new color content to the images. It is planned that by the use of the regularization in the proposed method, excessive blurring will not be a problem. Graphical display output of images will be shown in black and white for printed publication, but will be identified as "RGB" or by a particular color so as not to be misinterpreted. The method will recombine each color matrix into a finished denoised RGB image. Some prior works utilize images in gray scale format. In Figure 1.1 a color image and its individual color pixel intensity histograms for red, green and blue matrices are shown, and then a gray scale conversion of the same image and its histogram are show below. Notice the pixel intensity peaks are not the same for all colors. This demonstrates that each color matrix is in itself a unique image that will be affected uniquely by the same noise profile and subsequently the denoising process. The noise may be more pronounced in one color matrix than another. For this reason this research is to be carried out on each color matrix separately.

Denoising images is one of the first image processing steps, because without removing noise, a digital image is less useful for its intended purpose and other image processing steps will be less effective. Noise is defined as any unwanted disturbance in image data that is due to limitations in the sensing, signal digitization, or data recording process [40]. Noise in an image is similar to a weed in a garden. Weeds are said to be a plant in the wrong place. This holds true for noise as well. A noisy pixel in an image is in this sense an out of place pixel. Noise manifests itself in a digital image's pixel intensity values as outliers for a region of the image.

To understand noise we must be able to see it, know how it comes to exist, and how it can be detected. The effect of noise upon an image should be observable or at least measurable at the level used as a basis to warrant denoising. The general causes for noise creation should be understood. The methods to detect noise, mathematically or visually, should be defined. There are many types of noise, and many mechanisms at play in the capturing of a digital image that explain the existence of noise as a result of physical phenomena, or the device capturing the image. Noise can be produced by the image sensor and circuitry of a scanner or digital camera. Image noise can also originate in film grain and in the unavoidable shot noise of an ideal photon detector. Image noise is an undesirable by-product of the image capture process and obscures the desired information [37].

Figure 1.2 provides a pixel level view of noise. The top row shows a cat face image with and without noise in a full size image. The bottom row shows a magnified view of the cat's left eye with and without noise. In the magnified view individual pixels are visible, as are clearly out of place noisy pixels. This demonstrates how noise disrupts an image. Notice the

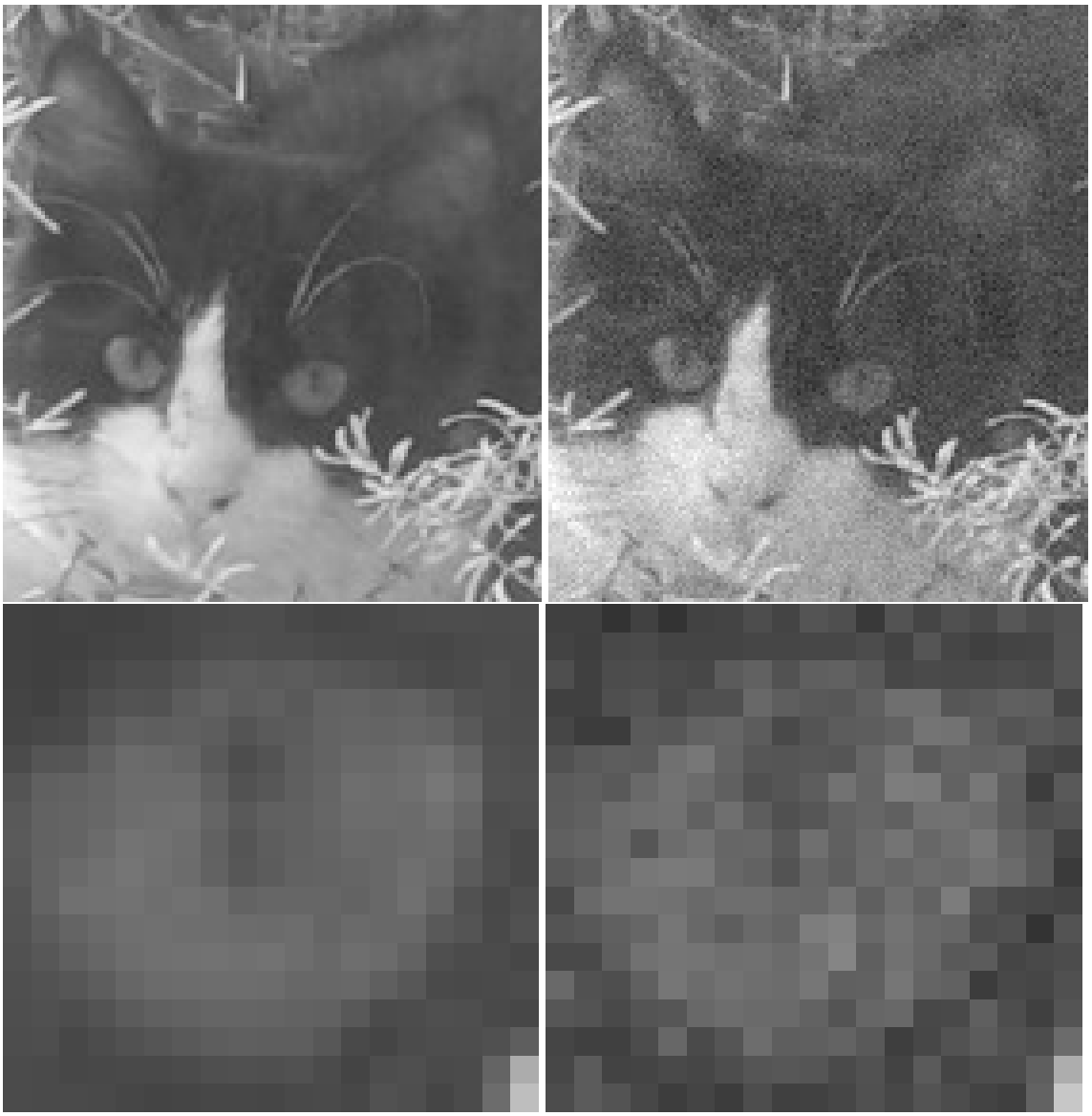

Figure 1.2: Cat Face Images Initial (top left), Noisy (top right) and Corresponding Cropped Views of the Left Eye Initial (bottom left) and Noisy (bottom left) with 10% Gaussian noise $\sigma^2 = 0.0011$

left images are without noise and the right images have noise. The level of Gaussian noise shown is 0 mean and variance 0.0011. Gaussian noise affects random pixels with intensity levels having magnitudes defined by the Gaussian function for the specified variance and mean. The result in the image is the color intensity of random pixels is higher than normal, which mottles regions having the same color, and disrupts edges and fine line detail.

In denoising experiments, measurable noise is artificially added to an existing image.

Table 1.1: Method Noise for Grayscale and RGB Images Shown As Standard Deviation, Variance and Percent Gaussian Noise

Grayscale	Grayscale	Grayscale	RGB	RGB	RGB
Std Dev σ	Var σ^2	$\pm$% G_{noise}	Std Dev σ	Var σ^2	$\pm$% G_{noise}
0.300	0.09	$\pm$30	0.010	0.100	$\pm$10
0.020	0.04	$\pm$20	$0.066\overline{6}$	0.0044	$\pm$6.6

This is called "method noise". A digital image is assumed to be smooth, and noise added to it makes it less smooth. However, some images a having a lot of edge detail may have an intensity profile that is far from smooth even before noise is added. Measurements of noise level and other quality measures are needed to optimize when to terminate denoising processes, and to measure effectiveness. There are many methods for creating noise to be added to an image to model a particular type of noise that may be found in real images. Gaussian Noise, Salt and Pepper Noise and Poisson noise are different types of noise. Gaussian noise is the type of noise used in most denoising experiments, and accordingly this research will use it as well. Gaussian Noise is additive noise and it is created by generating a Gaussian profile using a specified value for the variance σ^2. The amount of method noise is described by σ^2 as a number describing a portion of the total image variance to be added as noise. For gray scale images the method noise is applied to a single gray scale matrix. For RGB images only $\frac{1}{3}$ of the method noise is applied to each color matrix, so that when recombined to a RGB image the method noise is equivalent to that for the gray scale image. In this research Gaussian noise is implemented using the **MATLAB** function **imnoise(X,'gaussian',mean,var)**. All testing on individual color images will use a default mean of 0 and variance σ^2 values ranging from 0.0044 to 0.010, or equivalently stated is 6.6 to 10 percent, which corresponds to 20 to 30 percent applied to a gray scale image. One other way to describe method noise is to specify the amount of additional pixel intensity i_{pixel} added to pixels in the image matrix. Using this method our method noise range would be equivalent to adding from $\pm$20 to $\pm$30 to the pixel values of the image matrix. This is the method used to describe method noise in initial testing of the implemented PDE, see Guidotti [19]. Table 1.1 shows the correlation between method noise for gray scale and individual color images within the testing ranges used for this research.

Human vision is rather ambiguous to understand, in that we know what we see but don't always understand how we see it. This extends to human perceptions of image quality and subtle quality issues. Low levels of noise can be measured mathematically in an image yet

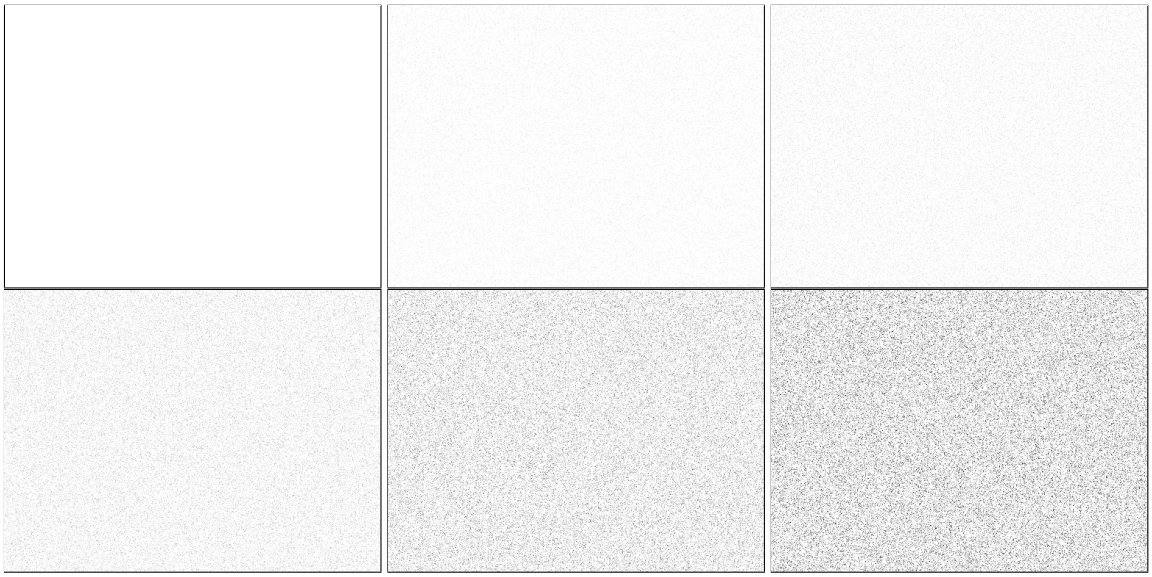

Figure 1.3: Various Levels Of Gaussian Method Noise In Isolation, $\sigma^2 = 0$, 0.025, 0.05, 0.10, 0.20, 0.30 (from top to bottom left to right)

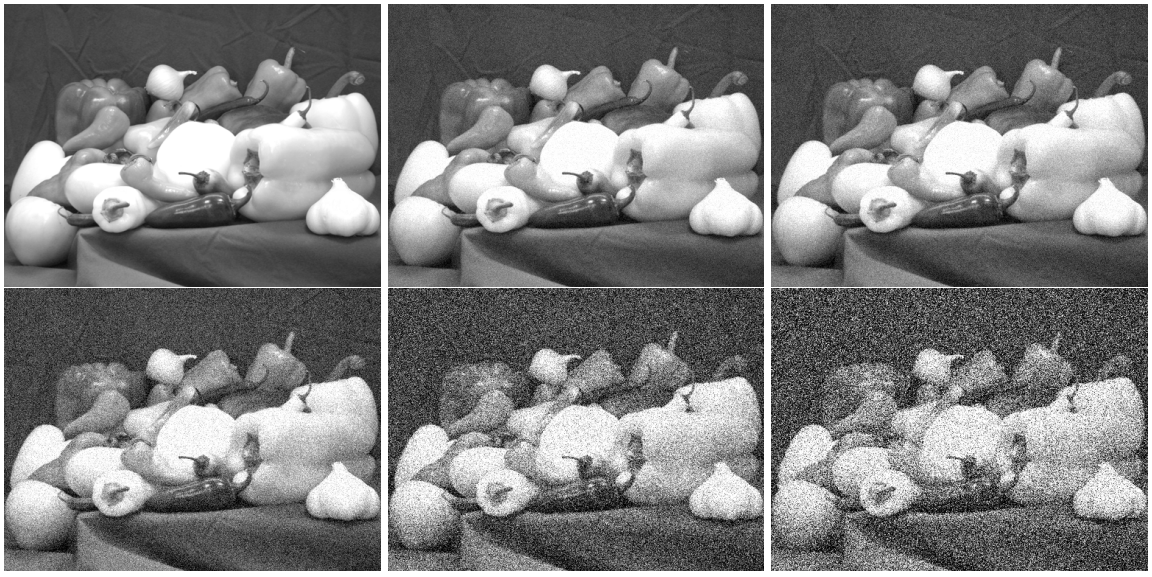

Figure 1.4: Various Levels Of Gaussian Noise Added To Peppers Image, $\sigma^2 = 0$, 0.025, 0.05, 0.10, 0.20, 0.30 (from top to bottom left to right)

may be undetectable, due to the compensating abilities of the human vision system. The lowest level of noise that is detectable by human vision is for $\sigma^2 \in [0.025, 0.10]$. Various values of method noise are shown in isolation on a blank image in Figure 1.3. In the top row, the leftmost image contains no noise and the remaining images in the row show increasing

noise of $\sigma^2 = 0.025$ and $\sigma^2 = 0.05$, both of which are very difficult to detect. The second row shows noise levels of $\sigma^2 = 0.10$, 0.20, and 0.30, which are more readily detected. Figure 1.4 demonstrates what those same levels of noise look like when added to the peppers image of the same size. It is notable that noise is easier to detect by visual observation in isolation than it is when combined with an image. Depending on the image, some levels of noise may not diminish the quality of what is seen and even large amounts of noise don't prevent the observer from gaining meaningful information from an image. For this reason it is critical to put into perspective quantitative measures of image quality, by considering the end use of the images being denoised.

In a very textured image containing many edges, noise can be harder or easier to observe. Noise may become indistinguishable from fine detail, or it might noticeably destroy fine detail. Ideally, denoising reduces noise to undetectable levels. Theoretically, denoising is carried out until improvement measures show no further improvement. Realistically, denoising terminates when a predefined threshold of quality is met. Practically, the threshold would correlate with a desired measured value depending on the eventual use of the image.

1.3 Improved Quality Measures Implemented

Typically, denoising research has relied on Peak Signal to Noise Ratio (**PSNR**) or Mean Square Error (**MSE**) to quantify the amount of noise removed from an image [25]. Both of these measures are easy to quantify, yet because they are reference based and global in form, they are impractical for denoising of real images. In addition they both have a low correlation to human perceptions of image quality [26]. A reference based measure assesses the difference between the initial ideal image without noise and the noisy image. Reference based methods, though meaningful for experimental work, are less practical for real world denoising problems, in which the noisy image is all that is available and the ideal uncorrupted image is sought after. Global measures quantify the entire image, such that each pixel is treated equally and so local variations in quality improvement or lack thereof are not as apparent. For these reasons an additional quality measure, the Structural Similarity Index (**SSIM**) will also be employed for quantification of denoising in this work. What follows is a brief presentation of each of the quality measures.

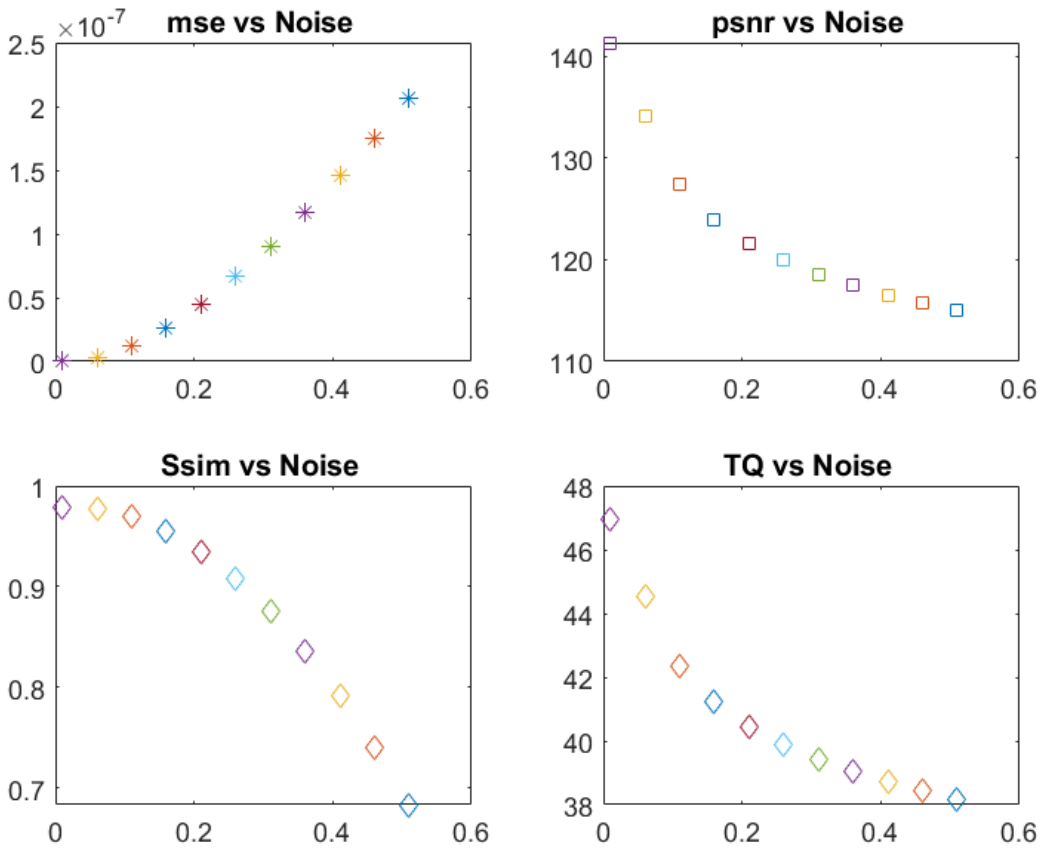

Figure 1.5: Quality Measures vs Noise Variance σ^2 including **MSE** (top left), **PSNR** (top right), **SSIM (bottom left**, and **TQ** (bottom right)

1.3.1 Mean Square Error (MSE)

Means Square Error (**MSE**) is the square of the difference in pixel intensity divided by image size. MSE is defined by the equation,

$$MSE = \Sigma_{i,j=0}^{N} \left[\frac{(u(i,j) - uorig(i,j))^2}{N^2} \right], \tag{1.1}$$

where N is the image side dimension, and u and $uorig$ are the current and unnoisy image matrices, respectively. Figure 1.5 shows how **MSE** reduces as noise level decreases. Typically the value of **MSE** is a number much less than one, but depending on image size, blocking scheme and ranges of pixel intensity, this value can be much larger than one.

1.3.2 Peak Signal to Noise Ratio (PSNR)

Peak Signal to Noise Ratio **PSNR** is defined as a logarithmically scaled ratio of the maximum pixel intensity and the **MSE** for the image. **PSNR** is defined by the equation,

$$PSNR = 10\log_{10}\frac{(max(u_0))^2}{MSE}, \tag{1.2}$$

where $max(u_0) = 255$ for color images. An examination of the formula shows that **PSNR** is a re-scaled version of **MSE**, referenced to the peak intensity values. It is reference based and also does not correlate well with human perceptions of quality. Typical values of **PSNR** could be between 20 to 100, but depend on image size and how much the image varies from the reference image. Figure 1.5 shows how **PSNR** increases as noise level decreases.

1.3.3 Perception Based Image Quality Evaluator (PIQE)

The Perception based image quality evaluator **PIQE** gives an image quality score that is based on several structural measures of the image allowing it to account for more localized variations in the image. **PIQE** is not reference based, so its assessment is only about the image in its current state. This measure shows better correlation to human perceptions of image quality. **PIQE** values ranges from 100 for worst to 0 for best. This measure is achieved using **MATLAB** command **piqe** [34]. During the refinement of the parameter optimization routine it was discovered that the **PIQE** measure, while generally informative about image quality, was not a good measure to use in the **TQ** score. This is because **PIQE** was not contributing consistently to the quality measure which resulted in triggering the termination criteria prematurely for the parameter optimization loops.

1.3.4 Structural Similarity Index (SSIM)

The Structural Similarity Index (**SSIM**) calculates a measure of structural similarity of the image matrix through statistical measures of contrast and luminance [26]. This measure is not reference based and offers a better correlation to human image quality perception and accounts for more localized features in the image. The **SSIM** value is implemented with the **MATLAB** command **ssim**$(u, uref)$. Typical values for this measure are numbers less than one, with a lower score signifying poorer quality. Figure 1.5 shows how **SSIM** increases as noise increases.

1.3.5 Total Quality Score (TQ)

To control the parameter optimization routine a combined assessment of image quality is needed to compare the change in image quality as the denoising progresses. For this purpose

a total quality score **TQ** is created. This value is calculated initially and at each denoising step; the proposed method will calculate a combined total image quality score using only three of these four measures previously discussed. Initially **PIQE** was included in the **TQ** score but its behavior in the denoising process proved troublesome as it had a curve that was not always consistently increasing or decreasing as the other measures did. Although **PSNR** and **MSE** are closely related and one actually utilizes the other, they vary in scale and so both are included. Figure 1.6 shows each quality measure as the denosing process advances.

This **TQ** measure will be utilized to direct termination of the denoising loop, as well as adaptive parameter adjustments to ensure productive progress is always towards improving the image quality. It is our expectation that this combined score will account for image specific characteristics and allow more localized treatment of images and will correlate better with observable image quality improvements. General comparison of end quality states of images compared to other research should use the raw quality scores in the comparisons and not this **TQ** score, as truly besides a relative measure for guiding parameter optimization it does not have a meaningful value on its own. In summary results for images and image blocks the final raw quality measures and percent changes will be shown to clearly identify the improved quality state. Thus the actual raw scores of each of the quality measures are shown in the tabular results at the block level. The measures are scaled by a factor of $\frac{1}{3}$ to keep the value of **TQ** less than 100.

$$TQ = 0.33 * MSE^* + 0.33 * SSIM^* + 0.33 * PSNR \tag{1.3}$$

The **TQ** measure has been refined to optimize the parameter optimization routine. The raw **PSNR** value, a relative **MSE*** value and a scaled **SSIM*** value form the **TQ** value as described by the formula below. Figure 1.6 shows how **TQ** and the other quality measures vary with increasing time and decreasing noise level as an image is denoised. A relative **MSE*** is used to ensure the measure contributes to the quality score consistently. This relative value is defined as follows,

$$MSE^* = MSE0 - MSE, \tag{1.4}$$

where **MSE0** is the **MSE** of the image in its initial noisy state at time $t = 0$. When quality is increasing, **MSE*** is positive and when quality is decreasing **MSE*** is negative. This relative measure is more appropriate for real world denoising, in which the unnoisy image is not known and **MSE*** is calculated from the current image and the initial noisy image. A scaled **SSIM*** score is used to ensure it is in the neighborhood of the other values, because **SSIM** values are small while the other values are much greater than one. Scaling it up

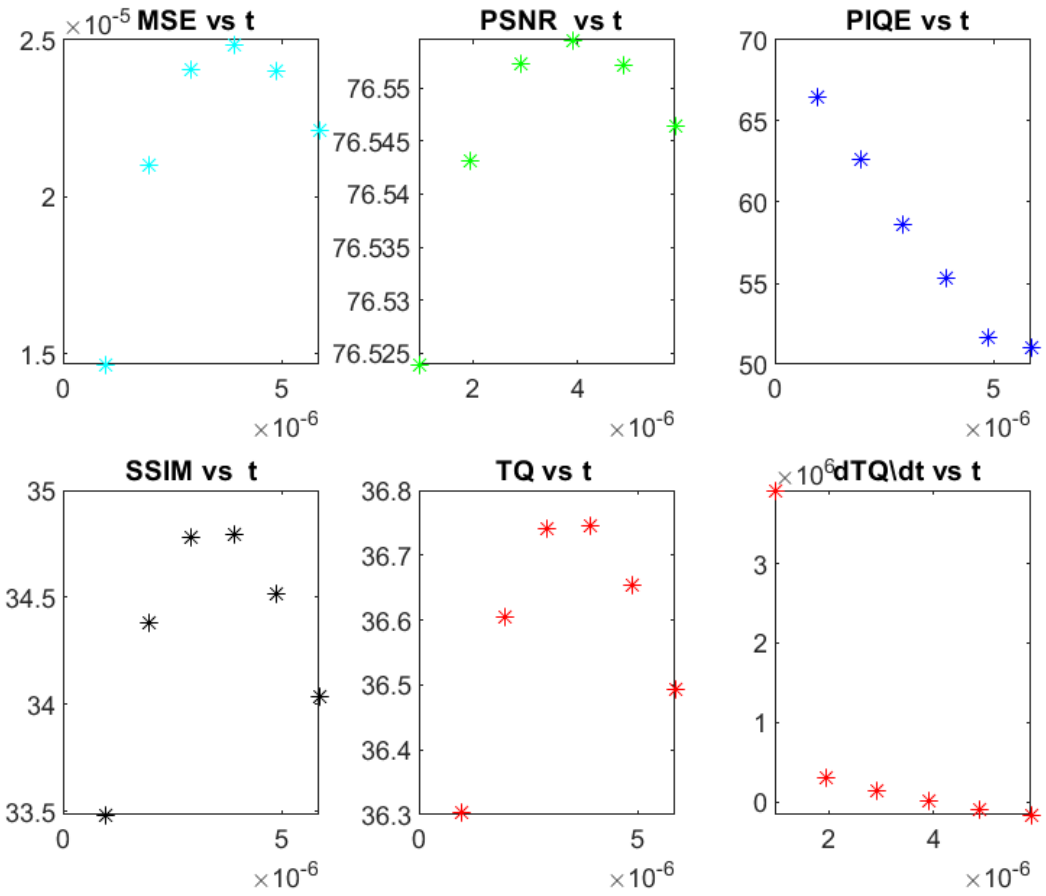

Figure 1.6: Quality Measures vs Time In Seconds For A **TQ** Optimization,Including: MSE, PSNR PIQE (top left to right) and SSIM , TQ and dTQ/dt (bottom left to right)

ensures the contribution to the **TQ** remains significant in comparison to the other measures. The scale factor is the average value of the initial values of the other two measures and is defined as $scaleSSIM = \frac{(PSNR0+MSE0)}{2}$, and is applied such that

$$SSIM^* = scaleSSIM \times SSIM. \tag{1.5}$$

1.4 Comparable Denoising Methods

To show the relevance of the proposed method, seven "state of the art" denoising methods will be used for comparative purposes in experiments. These methods represent a subset of the available methods in use for denoising images. Denoising methods can be cast into general functional categories including: averaging, filtering, transforming, variational, machine learning and PDE based diffusion methods, though some methods may span one or more categories. Neither variational nor machine learning methods will be included in these

comparisons. Figure 1.7 shows an image block denoised by seven comparatve methods that will be implemented in experiments. All but three of these methods are implemented with **MATLAB** commands using the image processing toolbox [23,24]. The remaining methods are achieved by running the implemented PDE in equation (2.8) used in this research with specific parameter values to effect regularized Perona-Malik AD, Gaussian Smoothing and the original Perona-Malik AD.

Averaging is achieved by averaging the pixel values in a sliding neighborhood within the image to reduce the outlier effects of noise in comparison to the other pixels in the neighborhood. One averaging denosing method, Median Filter, uses a one-dimensional neighborhood to average within, and is implemented with the **movmedian** command and signified by the abbreviation **MMV**. Filtering based denoising uses a threshold above or below which the image signal is modified to reduce noise. Three filtering methods achieve denoising using a low-pass filter to cut off or remove the portion of the signal that exceeds the threshold value. Threshholding with a hard and soft setting is implemented with the command **wthresh**, with a hard or and soft parameter setting. Hard threshsholding is a cruder application that truncates the eliminated signal while soft threshholding is more refined and does a linear blending across the eliminated signal. These two methods are abbreviated as **MTHh** and **MTHs** respectively. Weiner Filtering is a spectral filtering method and is implemented **mwiener2** and is abbreviated as **MW**. Smoothing is achieved by convolving a Gaussian Kernel with the image to blur noise away out of the image signal. Two Gaussian smoothing comparative methods are implemented. One is a **MATLAB** command **conv2** which is abbreviated as **MS**.

Anisotropic diffusion (AD) based denoising is achieved in two ways. The first with the implementation our proposed equation (2.8) which will be introduced in Section 2.7, as **KSSFast** also identified as **KSS** with parameters k and ε set by parameter optimization. The second AD denoising is achieved by using the **MATLAB** implementation Diffuse Filter, using the command **imdiffusefilt** which is identified as **IMDiffuse** or abbreviated **IDF**. Figure 1.7 shows a block of the BiloxiShrimpBoat image desnoised by seven of the comparative methods. In this figure the **KSS** image is the only implementation of the PDE in equation (2.7) shown. This is because the implemented code solves equation (2.7) using specific parameter settings for k and ε, which is this case defined the regularized PDE. Based on those settings **KSSFast** denoises the image accordingly, after which the final image is fed to the comparative method routine to generate results for same original noisy image.

Our proposed method is a PDE-based anisotropic diffusion method employing a fractional derivative that has a regularizing effect. Yet it can also be viewed as Spectral Domain Filtering because it utilizes Fourier Transform pairs. Localized focus is achieved through

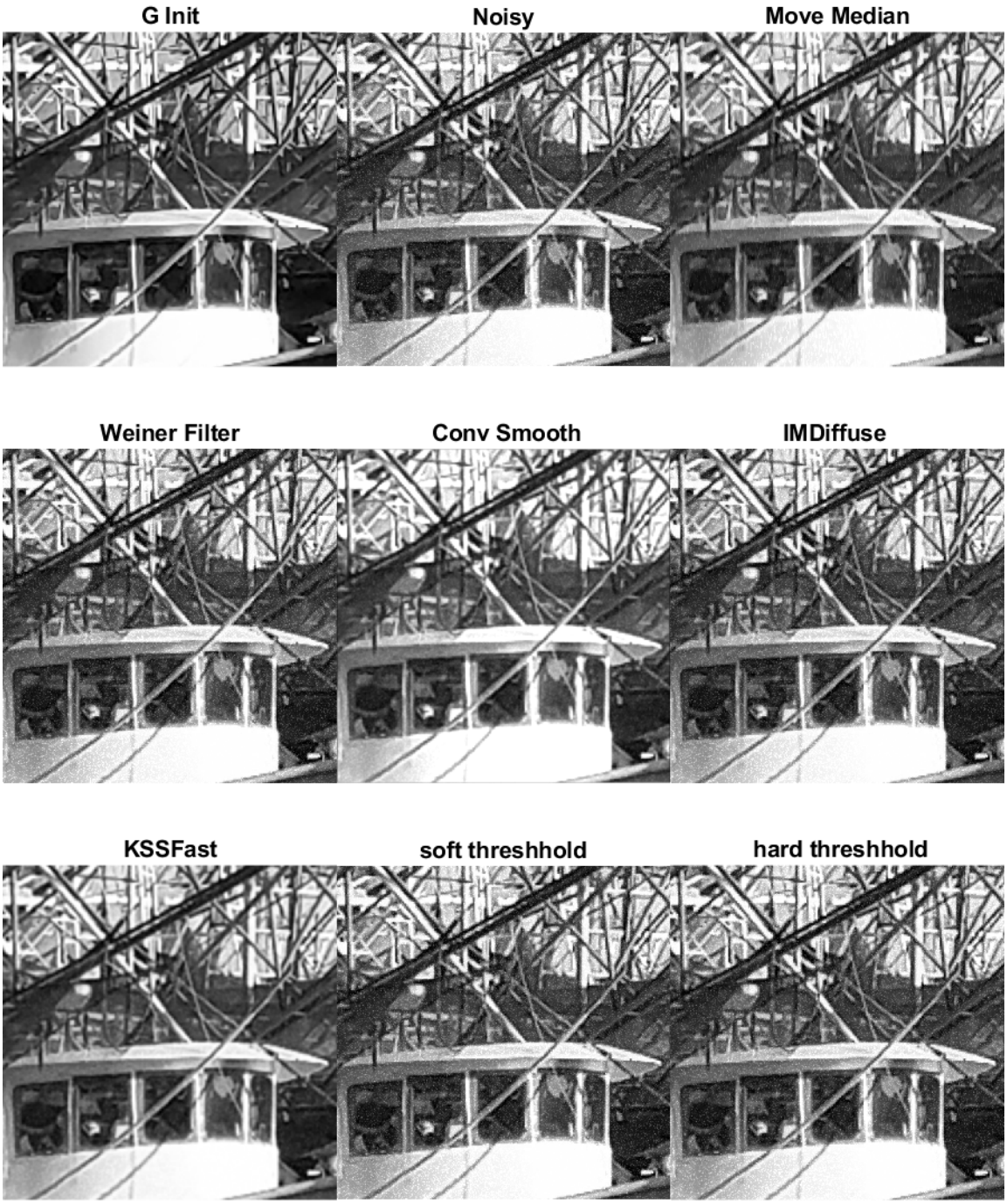

Figure 1.7: Boat Initial, Noisy and Denoised images for Green Block 22 shown are: Initial, Noisy, and Denoised By Moving Median (row 1 left to right), Weiner Filter, Convolutional Smoothing, Diffuse Filter (row 2 left to right), and KSS, Soft Thresholding, and Hard Thresholding (bottom left to right)

blocking images and separating color matrices for individual treatment. Adaptive parameter tuning at this local scale we expect will maximize and maintain local structure, whereas processing an entire image in gray scale might lose localized characteristics. In our proposed method the contribution of Fourier coefficients from higher wave number frequencies is minimized because of the decay they experience as wave number increases, thus having a denoising effect in itself which may mimic transform domain filtering based methods. Considering all these complexities, a synergistic effect of overlapping beneficial denoising results is expected.

Chapter 2

Literature Review

Initial denoising methods evolved from averaging, smoothing and then edge sharpening methods. This literature review provides a brief evolutionary perspective on these early methods with an emphasis on the development of anisotropic diffussion partial differential equations (AD PDE) methods. Some of these other methods are addressed in more detail in Chapter 3 as they are used to benchmark the proposed research.

2.1 Averaging

Median Filtering was the simplest and earliest denoising method [41]. By using an averaging formula it modified outlier pixel intensity values. Methods evolved from globally blind averaging of all pixels in an entire image into more localized and intelligent averaging that focused on and modified only outlier pixels within specified neighborhoods [37].

2.2 Smoothing

The Scale-Space method achieved smoothing by employing a convolutional filter, a Gaussian kernel convolved with the image [35]. This technique did effectively reduce noise, but it smoothed indiscriminantly which resulted in edge distortions, over blurring and image information loss [21]. These artifacts were to such an extent that secondary processing called segmentation was then required to recover lost edge detail. It was observed by Koenderink that the scale space method was comparable to modeling the process as solution to a heat equation or diffusion equation [42]. It is apparent now that his insight foreshadowed the advancements that followed in the form of artifact remedies. These approaches were PDE based segmentation methods to recover lost edge information by utilizing an edge detector. An edge detector could protect edges from over-smoothing yet allowed denoising in homogeneous regions. These methods utilized the heat equation or the diffusion equation to model the the image denoising process as an energy minimization problem.

2.3 Partial Differential Equation (PDE) based Methods

Initial Partial Differential Equation based methods included formulations modeled by the heat equation and the diffusion equation. The heat equation employed a constant conductance coefficient that acted upon the image on a global scale. Minimizing the thermal energy of a heated object was comparable to minimizing the pixel intensity of an image. Thermal energy flowed from hot to cold, evening out the object temperature profile. A denoised image pixel intensity profile similarly smoothed out and reduced sharp peaks.

The Diffusion Equation model began similarly with a diffusion coefficient γ, initially static and global, then evolved into a spatially varying, then spatially and temporally varying anisotropic nonlinear operator. A constant diffusion coefficient can not address local variations in an image with its global applied diffusion. This resulted in oversmoothing and edge distortions; the details of these detrimental effects are detailed in [20,21]. In order to avoid overprocessing edges and losing valuable image information, a more localized method to retain local features was achieved with spatially and temporally varying diffusivity coefficients. The isotropic diffusion equation is shown below with constant diffusivity.

$$u_t = \gamma \Delta u \tag{2.1}$$

The anisotropic diffusion equation shown below has a diffusivity coefficient that is spatially varying and constant in time.

$$u_t = \nabla \cdot (\gamma(u_0) \nabla u) \tag{2.2}$$

The diffusivity coefficient is spatially varying but constant in time because it is based on the initial image u_0 and evolved to be temporally variable through dynamic updating of the image as the denoising progresses [21]. The next step in increasing variability has a spatially and temporally varying anisotropic diffusion equation.

$$u_t = \nabla \cdot (g(|\nabla u|) \nabla u) \tag{2.3}$$

This form effectively disabled denoising near edges and enabled denoising in homogeneous regions. The diffusion action focused in regions without high gradients. This is a similar process to both the heat model and the noisy image problem. For the Heat equation this results in heat diffusion from hot regions to cold. For the noisy image problem this results in noise being removed from regions away from the edges where there are high pixel intensity gradients.

2.4 Perona-Malik Nonlinear Anisotropic Diffusion

Anisotropic diffusion coefficients vary with changes in the image matrix and allow spatially varying diffusion; the most notable AD denoising method was developed by Perona and Malik. Ironically it was born from these efforts not as a denoising tool, but as a secondary segmentation tool to recover lost edges after initial smoothing had removed noise and blurred edges. The Perona-Malik anisotropic diffusion equation offered a more localized treatment for images, with its spatially varying non-linear diffusion coefficient. Its effectiveness allowed it to replace convolutional smoothing methods as the primary denoising method [32]. A key element of the method was its edge detector, enabling retention of edge information. By employing a threshold coefficient k the edge detector could be varied to adjust the denoising and edge protecting processes to a particular image. In a sense it added one more degree of variablilty based on the adjusting magnitude of the gradient. The Perona-Malik equation offered two formulations in divergence forms, each with an edge detector that employed a gradient based diffusion coefficient k that achieved a more local focus [20].

$$u_t = \nabla \cdot [g(|\nabla u|)\nabla u], \tag{2.4}$$

where $g(|\nabla u|)$ is a gradient based edge detector taking two forms. The first edge detector has an exponential form shown below.

$$g_1(|\nabla u|) = e^{-(\frac{|\nabla u|}{k})^2} \tag{2.5}$$

The second edge detector had a reciprocal quadratic form shown below.

$$g_2(|\nabla u|) = \frac{1}{1+(\frac{|\nabla u|}{k})^2} \tag{2.6}$$

2.5 Perona-Malik Edge Detectors

Both the Perona-Malik edge detectors shown above utilize the image gradient. The gradient controls diffusion to act upon the image proportionally to its value at each image pixel location. The process of denoising or edge protection occurs when the gradient is either larger or smaller than the k threshhold parameter. Table 2.1 summarizes how k controls the direction of diffusion to achieve denoising or sharpening. When $|\nabla u| < k$ forward diffusion occurs resulting in denoising and when $|\nabla u| > k$ backward diffusion occurs resulting in sharpening as the edges are protected. This action can be achieved with a variety of edge detectors, though the gradient based edge detector was the first proposed. The magnitude of k depends upon the actual image and must be tuned to identify what is an edge effectively. When $k = 1$ the models become equivalent to Gaussian smoothing.

Table 2.1: Table 2.1 Edge Detector Controlled Diffusion with k Value

Edge Detector	Process	Diffusion
$\lvert\nabla u\rvert < k$	denoising	forward
$\lvert\nabla u\rvert > k$	sharpening	backward

The backward diffusion process is mathematically ill posed. This is apparent by carrying out the divergence after which the second term reveals how either or both the u_{xx} and u_{yy} coefficients can become negative and have a destablizing effect. This instability manifests itself in artifacts in the denoised image with staircasing and blockiness distorting what should be well defined edges. These issues generated attempts to perfect the equation or its edge detectors, to avoid artifacts. This included adjusting the order of the PDE, adding regularization or fidelity terms, and modifying the edge detector in a variety of ways [21,47]. Much of this work took the form of Perona-Malik variants with new edge detectors intended to control or delay the instability of the backward diffusion. Some methods offered different operators in the edge detector, employed regularizing coefficients, implemented higher order PDEs, or adding post processing steps to reduce the occurrence the artifacts. Of note is the research by Guidotti and Lambers, in which was proposed two well posed regularized variants of the Perona-Malik equation, of which one is implemented in this research [19,20].

2.6 Evolution of Edge Detectors

Edge detectors emerged in the PM equation initially as a gradient, but evolved into various forms including convolved gradients, Laplacians, and logarithmic combinations of those forms, as well as an additional regularizing coefficient all in an attempt to control the artifacts [47]. The well posed edge detectors proposed by Guidotti and Lambers had either a gradient or Laplacian as the edge detector operator, and each included a regularizing parameter that achieved a fractional derivative [19].

2.7 Regularizing Edge Detectors

These two well posed regularized Perona-Malik variants each were implemented with promising results in conjunction with a standard *GMRES* iterative solver for first order equations [17]. In both forms, regularization is achieved through the ε parameter which reduces the power to which either the gradient or Laplacian is raised. Fractional derivatives have proven effectiveness for regularization and for edge detection. Additionally, experiments using edge detectors having fractional derivatives have been shown to reduce or eliminate

artifacts, such as staircasing, which usually result from backward diffusion processes [27]. The gradient-based regularized form is shown below.

$$u_t = \left(\frac{1}{1 + k^2[(-\Delta)^{1-\varepsilon} u]^2} \right) \Delta u \tag{2.7}$$

This first form in equation (2.7) is implemented with Neumann boundary conditions and the noisy image as initial data, where k is the threshold parameter controlling the size of the Laplacian-based edge detector to distinguish edges, and ε is a regularizing coefficient to modulate the effects of uncontrolled growth of the Laplacian during the backward diffusion process. This equation modifies the original **Perona-Malik** equation in two ways. The first is that the ε creates a fractional derivative to be applied to the edge detector. The second is that the edge detector is Laplacian based instead of gradient based [15]. The second form, having a gradient based regularized equation is shown below.

$$u_t = \nabla \cdot \left(\frac{1}{1 + k^2|(\nabla)^{1-\varepsilon} u|^2} \right) \nabla u. \tag{2.8}$$

This second equation (2.8) is implemented with periodic boundary conditions and the noisy image as initial data, where k is the threshold parameter controlling the size of the gradient-based edge detector, and ε is a regularizing coefficient to modulate the effects of uncontrolled growth of the gradient during the backward diffusion process. This equation modifies the original Perona-Malik equation in one way. The ε achieves a fractional derivative in the gradient based edge detector [15]. It is worth mentioning that the periodic boundary conditions paired with this second equation, when applied to image deniosing, becomes difficult to implement without special handling [19].

In two recent implementations of diffusion based denoising, "wrap around effect" artifacts caused by periodic boundary conditions were successfully avoided [28,29]. Both utilized a buffering frame which extended the domain outside the image on all edges. These edge effect artifacts were manifested by the requirement that the image be the same at the boundaries at opposing domain edges. The domain buffer moved these ill effects out of the image and into the buffer. Post processing steps later removed the artifacts and the buffer region from the final result, but they still occurred. This "wrap around effect" artifact would, it is suspected, form a grid artifact over an image if it were denoised as separate sub blocks, having the artifact along every sub block boundary throughout the image. Additionally, implementing a domain buffer frame to remove these artifacts would add complexity to the sub block handling [27].

Both of these non-linear parabolic partial differential equations can be considered regularized improvements of the original Perona-Malik equation [19]. Lambers' testing

with both PDEs was implemented using gray scale images, for which the selection of the k and ϵ parameters was done through a trial and error approach, and employed a GMRES timestepping iterative solver. These equations offered significant improvements over the original PM equation. They were both well-posed and reduced the existence of artifacts in denoised images through the use of the regularizing coefficient [19]. Later testing with these equations used a Krylov Subspace Spectral (KSS) timestepping scheme in a fourth order implementation [27]. In this work as well all entire images were processed in grayscale, with parameters set by trial and error. Fourth order methods in combination with regularizing were found to have less artifacts but resulting in smaller time steps and longer processing times. The research with these edge detector forms investigated the one dimensional and later two dimensional effects of denoising with these edge detectors to varying degrees of success [22].

2.8 Denoising Literature Review Conclusions

Conclusions from this portion of the literature review are as follows. Image processing evolution has been driven by utilitarian needs and advancements, followed a heuristic path seeking to address deficiencies, with a focus on function before form, or effectiveness before mathematical rigor and theory, although the latter often may have explained the former. In particular, the Perona-Malik equation and its variants have dominated denoising methods for three decades. The evolution of edge detectors has progressed towards increasing local variability to increase denoising effectiveness. It is notable that in spite of its ill-posedness the Perona-Malik equation remains a persistently effective denoising method. For this reason new research to mitigate the resulting artifacts of ill-posedness of the Perona-Malik equation continues to be a viable research area. This phenomenon has been referred to as the Perona-Malik Paradox [21].

2.9 Motivation for Proposed Research

Work by Guidotti and Lambers with regularized edge detectors is significant and successful. This research combined many changes to the original Perona-Malik equation. These include a regularizing term, a modified edge detector, varied timestepping methods, trial and error parameter values and varied boundary conditions.The proposed research aims to continue along this research path utilizing only equation (2.7) applied to separate color image components, automated parameter setting, and efficient local image treatment through blocking images into smaller images that will be processed with block processing, using an evolved, simplified KSS time stepping scheme. In order to understand how this KSS method

is uniquely well suited for this image denoising problem, an overview KSS Methods and how they have evolved into the KSSFast implemented is provided in the following sections.

2.10 KSS Method Overview

Solution of variable coefficient parabolic PDEs for large systems has typically been done using explicit or implicit one-step or multistep methods. As the resolution of the system increases, these methods have requirements for stability or computational expense that leave them inadequate. Krylov Subspace Spectral Methods arose in 2005 [9] as an alternative timestepping scheme to solve parabolic variable-coefficient problems. KSS methods offer a stable, accurate, scalable and lower complexity solution. Time stepping utilizing spectral methods is ideal since the solution can be expressed as linearly combined basis functions that automatically satisfy the boundary conditions. These basis functions are approximate eigenfunctions of the spatial differential operator, allowing for individual evolution in time of each Fourier coefficient. Specifically KSS methods offer higher accuracy, stability and scalable solutions. The higher accuracy is achieved by computing each Fourier coefficient individually using an approximation that is optimal for it. Timestepping could be done using explicit, implicit or multistep methods with higher dimensional complexity and less desirable stability properties and restrictions on timestep size. It is for these reasons that KSS methods are used for timestepping, as other Krylov subspace methods such as of Hochruck and Lubich still have trouble with scalability [4].

2.10.1 Fourier Cosine Transform

The KSS Method is a spectral time stepping scheme which begins by applying the Fourier transform to obtain Fourier coefficients that appear in the solution. Since our PDE has a domain $[0,\pi]\bigotimes[0,\pi]$, such that the period $L=\pi$, with Neumann boundary conditions this spectral transformation is achieved with the two-dimensional Fourier cosine transform as defined below [49].

$$u(x,y,t_{n+1}) = \Sigma_{\omega_1=0}^{N}\Sigma_{\omega_2=0}^{N}\hat{u}(\omega_1,\omega_2,t_{n+1})\cos(\omega_1 x)\cos(\omega_2 y)\,dy\,dx, \tag{2.9}$$

where $u(x,y,t_{n+1})$ is the image matrix at time t_{n+1} and the Fourier cosine coefficient defined as

$$\hat{u}(\omega_1,\omega_2,t_n) = \frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi}\cos(\omega_1 x)\cos(\omega_2 y)u(x,y,t_n)dxdy. \tag{2.10}$$

In this research the continuous two-dimensional problem is implemented through a discretization in which evaluation of Fourier cosine coefficients is achieved by the two-dimensional discrete Fourier cosine transform (DCT2) of the image matrix at each time step. This is done by applying the one-dimensional discrete Fourier cosine transform as a composition, first along the rows for the x dimension, and then along the columns for the y dimension, and is implemented in **MATLAB** by the command **dct(dct(u))**. Similarly, after the solution is formed in the spectral domain it is returned to the spatial domain using the two-dimensional inverse discrete Fourier transform (IDCT2), which is achieved by applying the one-dimensional inverse twice, in a row-column algorithm by the command **idct(idct(u))** [23].

2.10.2 Fourier Coefficient Convergence

The Fourier Series for an L-periodic function will converge to the function at any point in the domain at which the function is continuously differentiable. The smoother the image function, the more rapidly the Fourier series will converge. If an image is p-times differentiable and its pth derivative is at least piecewise continuous, then the coefficients of the complex exponential form of the Fourier series satisfy the inequality shown below [14],

$$\hat{u}_{\omega_1} \leq \frac{C}{|\omega_1|^{p+1}+1}, \tag{2.11}$$

where C is some constant independent of ω_1. Thus as the wave number increases, the Fourier coefficients decay to zero as shown in Figure 2.1. We then extend this result for our PDE paired with Neumann boundary conditions for the denoising problem. The image matrix is assumed to be piecewise continuous in each dimension. It follows that the relationship shown in equation (2.11) can be extended to the two-dimensional Fourier cosine coefficients used in our implementation in a form shown below,

$$\hat{u}_{\omega_1,\omega_2} \leq \frac{C'}{|\omega_1\omega_2|^{p+1}+1}, \tag{2.12}$$

where the coefficient C' and the denominator would reflect the contribution from equation (2.11) for each dimension. Of importance is the decay rate of the Fourier coefficients to zero as the wave number increases. This justifies several simplifications to the KSS Method as applied to our denoising problem in the sections that follow and also allows simplification of the final form of the Lanczos recursion coefficients that are derived in Appendix A. At each timestep the Discrete two-dimensional Fourier cosine transform is applied to generate the Fourier coefficients, which are used in the KSSFast method, at the end of which the inverse Discrete Fourier cosine transform is used.

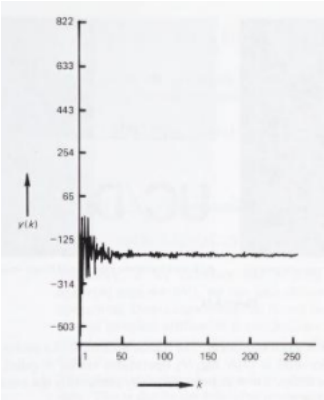

Figure 2.1: 1-D Fourier cosine coefficient decay with wave number

2.11 KSS Methods

KSS Methods begin with a spectral discretization of the PDE through an appropriate linear operator L; the PDE has the form $u_t = Lu$. The discretization matrix A arising from the spatial discretization of the operator L and converts the PDE into a system of ODEs. The problem solution u contains Fourier coefficients that can be expressed as a bilinear form using the standard inner product

$$\langle \mathbf{u}, \mathbf{v} \rangle = u^T f(A)\mathbf{v} \tag{2.13}$$

where

$$\mathbf{v} = \mathbf{u}(t_n), \tag{2.14}$$

is a spatial discretization of the solution from the previous timestep,

$$\mathbf{u} = \cos(\omega_1 \mathbf{x}) \cos(\omega_2 \mathbf{y}), \tag{2.15}$$

$f(\lambda)$ is a matrix function, and A is an $N \times N$ matrix arising from the differential operator L. This bilinear form enables approximation as a Riemann-Stieltjes integral which is ultimately evaluated with Gaussian quadrature. It is at this point where KSS Methods differ from prior timestepping methods. KSS Methods originally evolved to address timestepping solution methods for variable coefficient problems in a more efficient manner. KSS methods use the standard Lanczos or Block Lanczos algorithms to determine the nodes and weights for the quadrature rule, depending on the relationship between the vectors $\mathbf{u}$ and $\mathbf{v}$, in the bilinear form. In cases in where $\mathbf{u} = \mathbf{v}$, or $\mathbf{u}$ is very close to $\mathbf{v}$ use the standard Lanczos algorithm, otherwise the Block Lanczos algorithm is used [6,12].

Quadrature rules applied to cases in which $\mathbf{u} \neq \mathbf{v}$ result in negative weights and lead to instability. Several approaches have sought to mitigate this instability; most have involved

utilizing two applications of the standard Lanczos algorithms which are then combined for a quadrature rule. Lambers offered the Block KSS method as a highly accurate method in which a Gaussian quadrature formula approximates the bilinear form found in the Fourier coefficients of the solution [5,9]. The Block KSS method is characterized by its quadrature rule which is a direct result of the matrix A having real non negative eigenvalues. This enables the bilinear form to be seen as a Reimann-Stieltjes integral allowing approximation by Gaussian quadrature [2,7]. The ultimate evaluation of the quadrature utilizes Fourier Coefficients. Lambers' initial work on the Block KSS Method has been improved upon to increase efficiency while retaining its strengths of accuracy and implementation simplicity. Three significant simplifications related to methods used to arrive at the nodes and weights for the Gaussian quadrature and how the matrix exponential function is evaluated have brought the KSS Method to new heights of efficiency [6,12,45]. The sections that follow will develop the Gaussian quadrature rule as well as highlight the simplifications which improved the KSS Methods' efficiency.

2.11.1 Reimann-Stieltjes Integral equivalent to the bilinear form

The bilinear form is treated as the Reimann-Stieltjes integral ,

$$\mathbf{u}^T f(A)\mathbf{v} = \int_a^b f(\lambda) d\alpha(\lambda), \tag{2.16}$$

where a and b are the smallest and largest eigenvalues of A, enabling approximation techniques utilizing quadrature. This is due to the fact that the measure $\alpha(\lambda)$ is an increasing non-negative step function as long as $\mathbf{u} = \mathbf{v}$ or $\mathbf{u}$ and $\mathbf{v}$ are sufficiently close. Equation (2.18) [2,3] describes how the magnitudes of $\alpha(\lambda)$ at each step are the products of the coefficients of $\mathbf{u}$ and $\mathbf{v}$ in basis of the eigenvectors. Eigenvalues of A are real such that $b = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N = a$

$$\alpha(\lambda) = \begin{cases} 0 & \lambda < a \\ \Sigma_{j=i}^{N} \alpha_j \beta_j, & \mu \leq \lambda \mu_{i-1},\ \alpha_j = \mathbf{u}^T \mathbf{q}_j,\ \beta_j = \mathbf{q}_j^T \mathbf{v}. \\ \Sigma_{j=i}^{N} \alpha_j \beta_j, & \lambda \geq b, \lambda_j \leq \lambda < \lambda_{j-1} \end{cases} \tag{2.17}$$

2.11.2 Gaussian Quadrature instead of the Reimann-Stieltjes Integral

A Gaussian quadrature rule approximates the Riemann-Stieltjes integral and is ultimately used to approximate the bilinear form. In addition the use of the Gaussian quadrature rule is highly accurate for a 2-node quadrature rule and is exact for polynomials up to a degree 3. In general a K-node Gaussian quadrature rule is exact for polynomials up to degree $2K-1$

[2]. The Guassian quadrature rule is as follows,

$$\int_a^b p(\lambda) d\alpha(\lambda) = \Sigma_{i=1}^{K} p(x_i) w_i, \tag{2.18}$$

where the nodes x_i and weights w_i are obtained from the eigenvalues and eigenvectors of the τ_K matrix as described in Section 2.12 [3]. The nodes are the eigenvalues of the matrix τ_K, where K is the number diagonal blocks containing the series of recursion coefficients. The elements of the τ_K matrix are formed using the Lanczos recurrence coefficients, which are generated by the Lanczos algorithm as shown in Appendix A. The use of Gaussian quadrature results in a superior form of quadrature [2].

2.12 Lanczos Algorithm

For a K-node Gaussian quadrature rule the τ_K matrix will contain K α's and $K-1$ β's for the non-block case. The weights come from the first components of the eigenvectors as is shown for the non-block algorithm. The details of determining the quadrature nodes are described in Section 2.13. For a 2-node quadrature rule, the Lanczos algorithm generates the recursion coefficients α_1, α_2, and β_1. The matrix $\tau_K(\omega_1, \omega_2)$ is comprised of α_K coefficients on the diagonal and the β_K coefficients on the sub and super diagonals. These coefficients are needed to determine the eigenvalues of the τ_2 matrix, in order to evaluate the Gaussian quadrature rule. The subscript K denotes the number of non-block Lanczos iterations completed for **KSSFast** and also the number of diagonal entries in τ_K. If block Lanczos iterations are completed the subscript K denotes the number of iterations and also the number of diagonal blocks. **KSSFast** doesn't explicitly calculate the quadrature nodes which is why it is so fast.

2.13 Recent Simplifications to KSS Methods

KSS Methods use the non-block Lanczos algorithm when applied to the cases where $\mathbf{u} \neq \mathbf{v}$, but $\mathbf{u}$ and $\mathbf{v}$ are close enough to each other such that both the A and τ_K matrices are symmetric. Three simplifications to the KSS Method regarding the block algorithm through spectral analysis at high frequencies culminates in a faster, more efficient version referred to as **KSSfast**. The decay of Fourier coefficients discussed in Section 2.10 is used as a basis for these simplifications that allow reductions in computational expense and gains in efficiency. The next two sections explain the highlights of these simplifications.

2.13.1 Block Lanczos Matrix Decoupling of the τ_K Matrix

In early works Lambers offered an effective alternative to generate a block quadrature algorithm using concatenated vectors **u** and **v** in the bilinear form [5,9]:

$$\begin{bmatrix}\mathbf{u} & \mathbf{v}\end{bmatrix}^T f(A) \begin{bmatrix}\mathbf{u} & \mathbf{v}\end{bmatrix}, \tag{2.19}$$

which results in a 2×2 matrix form of the quadrature rule,

$$\int_a^b f(\lambda) d\mu(\lambda) = \begin{bmatrix}\mathbf{u}^T f(A)\mathbf{u} & \mathbf{u}^T f(A)\mathbf{v} \\ \mathbf{v}^T f(A)\mathbf{u} & \mathbf{v}^T f(A)\mathbf{v}\end{bmatrix}, \tag{2.20}$$

where $\mu(\lambda)$ is a matrix function of λ, each entry is a measure with the same form as $\alpha(\lambda)$ as described in equation (2.19) [3]. The quadrature rule is as follows,

$$\int_a^b f(\lambda) d\mu(\lambda) = \Sigma_{j=1}^K f(t_l)\mathbf{v}_j\mathbf{v}_j^T, \tag{2.21}$$

with scalar nodes t_j, and the associated weights are the 2×2 matrices $\mathbf{v}_j\mathbf{v}_j^T$, and where $\mathbf{v}_j$ is a $2-$vector of the first two components of each eigenvector of τ_K. The method to obtain the nodes t_j and vectors $\mathbf{v}_j$ utilizes orthogonal matrix polynomials and uses the Lanczos recursion coefficients found in the block tridiagonal matrix τ_K, arising from the Lanczos algorithm [2]. This matrix is shown below for the $K = 2$ case.

$$\tau_2 = \begin{bmatrix}M_1 & B_1^T \\ B_1 & M_2\end{bmatrix} \tag{2.22}$$

The block Lanczos algorithm is simplified through asymptotic analysis by Palchak et al., which shows that the blocks forming the τ_K matrix approximately decouple. Specifically, as $\omega \to \infty$ the Fourier coefficients occupying the off diagonal entries in M_K, B_K and B_K^T decay to zeros, as described in Section 2.10 and in [6,9] as shown below.

$$B_K(\omega) = \begin{bmatrix}B_K(1,1) & B_K(1,2) \\ B_K(2,1) & B_K(2,2)\end{bmatrix} \longrightarrow \begin{bmatrix}B_K(1,1) & 0 \\ 0 & B_K(2,2)\end{bmatrix}, \tag{2.23}$$

and

$$M_K(\omega) = \begin{bmatrix}M_K(1,1) & M_K(1,2) \\ M_K(2,1) & M_K(2,2)\end{bmatrix} \longrightarrow \begin{bmatrix}M_K(1,1) & 0 \\ 0 & M_K(2,2)\end{bmatrix}, \tag{2.24}$$

where the diagonal elements at indices (1,1) and (2,2) are all that remain in these sub blocks after the Fourier coefficients decay at high wave numbers. Of interest is that the odd rows contain frequency dependent entries and the even rows contain frequency independent entries, warranting a shuffling of rows to separate the matrix accordingly. We use a permutation

matrix to reorder the odd rows and columns, and even rows and columns, and decouple the τ_2 matrix to the form shown below.

$$\tau_2 = \begin{bmatrix} X_{fd} & 0 & X_{fd} & 0 \\ 0 & X_{fi} & 0 & X_{fi} \\ X_{fd} & 0 & X_{fd} & 0 \\ 0 & X_{fi} & 0 & X_{fi} \end{bmatrix} \longrightarrow \begin{bmatrix} X_{fd} & X_{fd} & 0 & 0 \\ X_{fd} & X_{fd} & 0 & 0 \\ 0 & 0 & X_{fi} & X_{fi} \\ 0 & 0 & X_{fi} & X_{fi} \end{bmatrix}. \tag{2.25}$$

This simplification of the τ_2 matrix causes two desirable effects. The first is the decoupling of the Block matrix into a τ_{2a} matrix containing frequency dependent components and a τ_{2b} matrix containing frequency independent components. A simplified quadrature rule requires two applications of non-block Lanczos, one for the τ_{2a} matrix and one for the τ_{2b} matrix instead of a single Block Lanczos algorithm applied to the τ_2 matrix. In these applications of non-block Lanczos the initial vector used for τ_{2a} is $\mathbf{u}$ and for τ_{2b} is $\mathbf{v}$. The section following explains how this last application to τ_{2b} makes a negligible contribution to the quadrature.

$$\tilde{\tau}_2(\omega) \tilde{=} \begin{bmatrix} \tau_{2a} & 0 \\ 0 & \tau_{2b} \end{bmatrix}, \tag{2.26}$$

where $\tilde{\tau}_2$ signifies the permuted new form of τ_2 and $\tilde{=}$ indicates that this new matrix is approximately equal to its prior form, in the limit as wave number tends to infinity.

2.13.2 Reduced Expense Quadrature and Matrix Exponential

The second major effect of the simplification to the τ_2 matrix was recognized by Bingham in her examination of the sub blocks shown in equations (2.25) and (2.26). The τ_2 matrix is a 4×4 matrix and has four eigenvalues. It should be noted that the τ_{2a} matrix has maximal eigenvalues λ_1 and λ_2 as is referenced by the subscript (1,1) in equations (2.24) and (2.25). Similarly the τ_{2b} matrix contains has lesser eigenvalues λ_3 and λ_4 as is referenced by the subscripts (2,2) in equations (2.25) and (2.26). Because λ_1 is the largest eigenvalue, its contribution dominates the evaluation of the matrix exponential, and so the contributions from λ_2, as well from λ_3 and λ_4 from the τ_{2b} matrix are justifiably neglected [45]. The τ_{2b} matrix is not needed becuase only the (1:2,1:2) block of $e^{\tau_2 \Delta_t}$ is needed, and τ_{2b} contributes nothing to that. The τ_K matrix is effectively reduced from a 4×4 matrix to a 2×2. This reduces the computational expense of the KSS Method in half, in that the non-block Lanczos algorithm need only be performed once on the τ_{2b} matrix. Additionally, and more significantly, the computational expense is halved in the evaluation of the matrix exponential at each timestep, resulting in significant time savings, and so this version is referred to as **KSSFast**. In the implementation this revises the matrix exponential evaluation from

$$\mathbf{u}^T f(A)\mathbf{v} \tilde{=} [e^{(\tau_2 \Delta t)}]_{1,1} \mathbf{u}^T \mathbf{v} \tag{2.27}$$

to become after decoupling

$$\mathbf{u}^T f(A)\mathbf{v} \tilde{=} [e^{(\tau_{2a}\Delta t)}]_{1,1}\mathbf{u}^T\mathbf{v} \tag{2.28}$$

.

2.14 KSS Literature Review Summary

In summary, the KSS method has evolved considerably since initially described by Lambers in [5]. In that work the stage was set for later research to make 2 major simplifications that modified KSS methods. Work by Palchak achieved more than a fifty percent reduction in computational expense while work by Bingham effected a more efficient approximation of the exponential function [6,45]. The KSS Method temporal accuracy is $O(\Delta t^{2K-1})$ for parabolic problems, and under appropriate assumptions on coefficients with $K = 1$ for a single node KSS method is unconditionally stable. KSS methods compute a single Krylov subspaces at each timestep. This proposed research will apply this most recent form used in Bingham's research, **KSSfast**.

2.15 Applicability of KSS Method to the proposed Anisotropic Diffusion PDE

In his 2013 work, Lambers applied KSS methods to the AD denoising problem to offer a more suitable time stepping scheme that offers speed and accuracy [28]. Explicit methods are not practical due to constraints requiring small time step size. Other alternatives such as backward Euler time-stepping combined with iterative methods such as MINRES require a large number of iterations and still have less than effective results due to artifacts associated with high frequency oscillations. For problems having Neumann boundary conditions KSS methods are well suited since solutions can be evolved in time using approximations of the solution operator that are optimal for each coefficient. A recent AD denoising paper compared the performance of KSS, finite element, finite difference and an operator splitting method performance. In the results summary KSS methods are described as a "best of both worlds" approach because they offered explicit method efficiency and implicit method stability allowing higher order accuracy in time and favorable stability properties [50]. The application of KSS methods, specifically **KSSFast**, to image denoising has a unique opportunity to apply a spectral method to a truly spectral application. Color images are distinguished by color pixel intensities that in a denoising process are color signals decaying with time. There may still be other opportunities to optimize the problem in the spectral domain utilizing the decay of Fourier coefficients. It is hypothesizied that this work may yield revisions of the method applied to this problem in terms of more efficient

solutions using parameter refinement and possibly more efficient timestepping. Since the Anisotropic Diffusion PDE used in this research has a variable coefficient containing a fractional derivative this problem is well suited to be solved by KSS methods because the spectral discretizations are very compatible to evaluating fractional derivatives in the spectral domain.

Chapter 3

Proposed Method

This chapter will describe in detail efforts to investigate the image denoising problem; including these components:

- The regularized Perona-Malik equation variant presented by Guidotti shown in equation (2.7).
- Adaptive parameter refinement to tune denoising automatically without human intervention to maximize image quality.
- Testing on color high resolution digital images.
- Comparison of denoising effectiveness against state of the art denoising methods for time and final quality state.
- Blocking images for efficient parallel processing.
- Use of KSSfast, a simplified and efficient version of KSS timestepping.

It is the intent of this research to closely examine Guidotti's first equation (2.7). This equation distinguishes itself by using a Laplacian based edge detector, that we expect will allow a smoother treatment of the image to reduce noise. In this work the PDE has the same form as in [19], but the method in which it is implemented has more localized treatment in that it is applied to each of three color matrices individually, and to smaller sub blocks of the images, with adaptive parameter setting. It is expected that the individual color matrix and sub block treatment will result in more effective and efficient denoising. A new quality based automatic method will be employed to tune each of the three denoising parameters for each image, including k for edge detection, ε for regularization, and dt for timestepping. This will be achieved through experiments on individual color images, or smaller sub blocks thereof, while utilizing a more expansive image quality assessment scheme to tune the denosing parameters. The intent is to gain a more localized approach while automating parameter setting in an adaptive way.

3.1 Modeling The Noisy Image Problem

A noisy image is created in the real world as the digital image is captured. The denoising process aims to remove as much noise as possible from the corrupted image. So, in reality there is no ideal unnoisy original image that is known. There is only the current noisy state and the expectation that there could be a more ideal quality image with denoising success. In denoising experiments a noisy image is created by intentionally adding noise to an ideal image. In this way, the amount of noise is controlled and the success of denoising can be measured for the experiment. The creation of the noisy image then is modeled by $u_0 = u_{orig} + n_{\sigma^2}$, to create the initial image u_0 beginning with the ideal image u_{orig} and adding Gaussian white noise is which is quantified by a variance (σ^2) of the pixel intensity. This variance value determines the **method noise**. Method noise is measurable as $n_{\sigma^2} = |u_0 - u_{orig}|$. The denoising process seeks to minimize the method noise at each successive timestep of the denoising process.

The reader should be aware that all testing for this research was carried out using **MATLAB** revision **R2020a**, and all commands for such will be identified in bold throughout this document. The initial image noise is read into the algorithm with the **MATLAB** command $u = \textbf{imread}(image.fmt)$, where the file format designation $.fmt$ may be one of several formats including $.jpg$ or $.png$. Any size or shape image can be used, but before any processing each image will be cropped to be a square $N \times N$, which ensures a square image matrix, based upon the user specifying a block side dimension to be used.

The noise is created in the form of a noisemask having the same dimension as the test image sub blocks to which it will be added. This noisemask is a matrix having all zero pixel intensities except where noise has increased them to the specified Gaussian values. The noisemasks for this research were created by adding Gaussian noise to each initial single color image sub block using the command **imnoise**(u,'gaussian',mean,var), where $mean = 0$ and variance $\sigma^2 \in [0.0044, 0.010]$. After adding noise to the image in the **uint8** format it is then converted to **double** format to ensure accuracy.

3.1.1 Partial Differential Equation

The image denoising process is modeled with the PDE developed by Guidotti and Lambers as shown previously in equation (2.7). This is a regularized variant of the original Perona-Malik nonlinear anisotropic diffusion model having Neumann boundary conditions. This model shown below will be implemented with **KSSfast** time stepping which will advance an initial noisy image towards a more optimum denoised image.

$$\begin{cases} \mathrm{u}_t = \frac{1}{1+k^2[(-\Delta)^{1-\varepsilon}u]^2}\Delta u & \text{in } \Omega,\ t>0, \\ \delta_v u = 0 & \text{on } \delta\Omega,\ t>0, \text{ Neumann boundary conditions} \\ \mathrm{u}(0) = \mathrm{u}_0 & \text{in } \Omega,\ t=0, \text{ initial data.} \end{cases}$$

As this PDE is applied to the image denoising problem the domain, boundary conditions and initial data have significant meaning. Since all images processed will be square $N \times N$, the PDE domain Ω is also a square domain $[0,\pi]\bigotimes[0,\pi]$. The Neumann boundary conditions require that the partial derivative of the solution $\delta_v u = 0$ on domain boundary $\delta\Omega$, which simplifies the form of the solution to only contain Fourier cosine coefficients. The initial data u_0 is the initial noisy image. Additionally the Neumann boundary conditions impose upon the denoised image a guarantee that there will be a smooth transition of the image at its boundary and also that no edge or corner artifacts are expected to occur. It may be of interest to see the discussion about artifacts to learn the ill effects that can occur due to having periodic boundary conditions which can be found in Section 2.8. For brevity the edge detector function may be denoted as $g(u) = \frac{1}{1+c^2[(-\Delta)^{1-\varepsilon}u]^2}$ so that the PDE has the form $u_t = g(u)\Delta u$, in which the differential operator L is apparent in $u_t = Lu$. It should be noted that the PDE is non-linear because of its dependence on u in the Laplacian that resides in the non-linear coefficient $g(u)$, which is defined in Section 3.3.1.

3.2 KSSFast Algorithm

The **KSSFast** algorithm previously explained in Sections 2.11 to 2.13 is summarized here. Each Fourier cosine coefficient is a bilinear form involving a matrix function-vector product, which is treated as a Riemann-Stieltjes integral, which is approximated by Gaussian quadrature, using only the first components of the eigenvectors, thus enabling reduced computational expense. To apply **KSSFast** to our denoising problem, three steps are neccessary for the Gaussian quadrature rule. First, the Fourier cosine coefficients are obtained from the discrete Fourier cosine transform at each timestep. Second, the Lanczos recursion coefficients determine the eigenvalues used in a matrix exponential. Third, combining the results from these prior steps, an approximate matrix function vector product forms the solution at the next timestep. Each step is detailed in the following sections.

3.2.1 Fourier Coefficients

For this PDE our solution $u(x,y,t_n)$ contains only cosine terms because the Neumann boundary conditions simplify the exponential to only real valued exponential components. The discrete Fourier cosine transformation (DCT) is applied to the image matrix for each

dimension, effecting a two-dimensional discrete Fourier cosine transform yielding Fourier coefficients which are defined by the inner product

$$\hat{u}(\omega_1,\omega_2,t_{n+1}) = \langle \cos(\omega_1)\cos(\omega_2), u\rangle = \int_0^\pi \int_0^\pi \hat{u}(x,y,t_{n+1})\cos(\omega_1 x)\cos(\omega_2 y)\,dx\,dy. \tag{3.1}$$

The discretized solution in the spectral domain is as follows,

$$u(x,y,t_{n+1}) = \Sigma_{\omega_1,\omega_2=1}^{\infty}\hat{u}(\omega_1,\omega_2,t_{n+1})\cos(\omega_1 x)\cos(\omega_2 y), \tag{3.2}$$

where the basis functions $\cos(\omega_1 x)\cos(\omega_2 y)$ are chosen due to the Neumann boundary conditions, and ω_2 and ω_1 are integers. The Fourier coefficients, $\hat{u}(\omega_1,\omega_2,t_n)$ of the solution are obtained using the discrete Fourier cosine transform at each time step to get the matrix of the Fourier cosine coefficients as mentioned in Section 2.10,

$$\hat{u}(\omega_1,\omega_2,t_n) = \mathrm{C}(u(x,y,t_n)), \tag{3.3}$$

where u is the image matrix from the previous time step, and C is the two-dimensional discrete Fourier cosine transform.

Fourier coefficients defined by the inner product

The inner product is

$$\langle f(\Delta), v\rangle = \left\langle \frac{1}{\pi}\cos(\omega_1 x)\cos(\omega_2 y), e^{-L\Delta t}\tilde{u}(x,y,t_n)\right\rangle, \tag{3.4}$$

where ω_1 and ω_2 are integer valued wave numbers, in which the spatial domain is

$$\Omega = [0,\pi]\otimes[0,\pi], \tag{3.5}$$

and where the standard inner product on Ω is

$$\langle f(x,y), h(x,y)\rangle = \int_0^\pi \int_0^\pi f(x,y)h(x,y)\,dx\,dy. \tag{3.6}$$

The solution operator is $e^{-\lambda\Delta t}$, which when multiplied by the solution u, from the previous time step yields the solution at the next time step.

Bilinear form results from Spatial discretization of the Fourier coefficients

The bilinear form approximates the Fourier coefficients using the standard inner product:

$$\langle \mathbf{u}, f(A)\mathbf{v}\rangle = \mathbf{u}^T f(A)\mathbf{v}, \tag{3.7}$$

where

$$\mathbf{v} = \mathbf{u}(t_n), \tag{3.8}$$

is a spatial discretization of the solution from the previous timestep,

$$\mathbf{u} = \cos(\omega_1 \mathbf{x}) \cos(\omega_2 \mathbf{y}), \tag{3.9}$$

where

$$A = L_N \tag{3.10}$$

is an $N \times N$ nonsymmetric matrix arising from the discretization operator L, which has real eigenvalues such that $b = \lambda_1 \geq \lambda_2 \geq \cdots \geq \lambda_N = a$, and $f(\lambda) = e^{-\lambda t}$ is a matrix exponential function.

3.2.2 Eigenvalues of τ_k Matrix

In **KSSFast** the evaluation of the bilinear form $\mathbf{u}^T f(A)\mathbf{v}$ is achieved through the Gaussian quadrature rule in which the eigenvalues are calculated using the Lanczos recursion coefficients: α_1, α_2, and β_1, as described in Section 2.12 and derived in Appendix A. For $k = 2$, a $2-$node Gaussian quadrature rule requires calculation of all the eigenvalues of $\tau_{2a}(\omega_1, \omega_2)$, a SPSD matrix, as explained in Section 2.13. Since τ_{2a} is a 2×2 matrix computing the two eigenvalues, λ_1 and λ_2, uses the Lanczos recursion coefficients in formulas for the trace, discriminant, determinant in the formulas below for eigenvalues.

$$Tr = \alpha_1 + \alpha_2 \tag{3.11}$$

$$D = \alpha_1 \times \alpha_2 - \beta_2 \tag{3.12}$$

$$disc = \sqrt{Tr^2 - 4 \times D} \tag{3.13}$$

$$\lambda_1 = \frac{(Tr + disc)}{2} \tag{3.14}$$

and

$$\lambda_2 = \frac{(Tr - disc)}{2} \tag{3.15}$$

3.2.3 Approximation of the Matrix Exponential

Evaluation of the matrix exponential function is done at each time step. The simplification in Section 2.13 justifies evaluation of the exponential of τ_{2a} instead of the whole τ_2 matrix. The product of the eigenvalues and the timestep are exponentiated for each frequency in the $2-$dimensional spectral discretization.

$$[f(\tau_{2a})]_{1,1} = [e^{-\tau_{2a}\Delta t}]_{1,1}. \tag{3.16}$$

The matrix τ_{2a} is 2×2, and its matrix exponential is evaluated for each Fourier cosine component in two dimensions at each time step. This calculation is achieved using a divided difference formula proposed by Higham, which only requires the eigenvalues of A be known [49,15].

$$[f(\tau_{2a})(\omega_1, \omega_2)]_{1,1} = f(\lambda_1) + \frac{f(\lambda_2) - f(\lambda_1)}{\lambda_2 - \lambda_1} \times (A11 - \lambda_1), \tag{3.17}$$

where λ_1 and λ_2 denote the eigenvalues, $A11 = \alpha_1$, $f(\lambda) = e^{-\lambda \Delta t}$, and the subscript $(1,1)$ signifies that only the eigenvalues of τ_{2a} for frequency dependent terms are used in the exponential. Thus $[f(\lambda)(\omega_1, \omega_2)]_{1,1} = [e^{-\tau_{2a}\Delta t}]_{1,1}$ results in an approximation of the $(1,1)$ terms of $[e^{-\tau_{2a}\Delta t}]_{1,1}$ at reduced computational expense.

3.2.4 Solution is a matrix-function vector product

The solution is formed by multiplying the matrix exponential by the Fourier Coefficients to form the spectral solution.

$$\hat{u}(\omega_1 x, \omega_2 y, t_{n+1}) \tilde{=} [f(\tau_{2a})(\omega_1, \omega_2)]_{1,1} \times \hat{u}(\omega_1, \omega_2, t_n) \tag{3.18}$$

Finally the solution is returned to the spatial domain by applying the inverse two-dimensional discrete Fourier cosine transform

$$u(\omega_1 x, \omega_2 y, t_{n+1}) \tilde{=} \mathbf{C}^{-1}[[f(\lambda)(\omega_1, \omega_2)]_{1,1} \times \hat{u}(\omega_1, \omega_2, t_n)] \tag{3.19}$$

This completes the description of the KSSFast Method.

3.3 KSSFast Numerical Implementation

The implementation of the problem on the discretized domain requires development of the discretized Laplacian in the spectral domain where the fractional derivative achieved by the parameter ε is formed. The fractional derivative is well defined as discussed in [19]. The discrete Fourier inverse cosine transform is then applied to the fractional Laplacian to return it to the spatial domain, to form the edge detector function $g(u)$. Then $g(u)$ is incorporated into L in forming the Lanczos recursion coefficients which are incorporated into the solution by way of evaluation of the matrix exponential. The sections that follow describe these steps.

3.3.1 Discretized Solution induced by L

Using the linear operator, the original PDE to be solved as follows,

$$u + Lu = 0 \tag{3.20}$$

where the nonlinear operator is

$$Lu = g(u)\Delta u. \tag{3.21}$$

For each timestep a linearization of the orignial PDE is done by evaluating $g(u)$ at a fixed time t_n so that it becomes $g(u^n)$ depending only on x and y. To construct the solution for this problem, several components are needed: the discretization in the spatial and spectral domains, the spectral discretization of the fractional Laplacian $\widehat{\Delta_N}$, and finally the spectral discretization of the edge detector $g(u^n)$ must be determined.

Spatial and Spectral Discretizations

This edge detector $g(u^n)$ is constructed in the spectral domain, as the DCT the discretization mesh from the spatial to the spectral domain. The spatial and spectral discretizations are isometric because all images use will be square, as are any sub blocks of the image. The spatial discretization mesh is described by

$$\Omega_{x,y} = [0,\pi] \otimes [0,\pi], \tag{3.22}$$

The spectral discretization mesh is described by

$$\Omega_{\omega_1,\omega_2} = [0,1,2,\ldots,N] \otimes [0,1,2,\ldots,N], \tag{3.23}$$

where N is the side dimension of the square image .

Spectrally Discretized Laplacian $\widehat{\Delta}_N$

The discretized Laplacian is formed by using a discrete Fourier Cosine transform to transform the image from the spatial domain to the spectral domain and generates the Fourier cosine coefficients. Then the Fourier cosine coefficients are multiplied by a spectral discretization constant $\hat{c}_{\omega_{xy}}$ which consists of the sums of the squares of frequencies: $\hat{c}_{\omega_{xy}} = (\omega_{x,N}^2 + \omega_{y,N}^2)$. The negative spectral Laplacian has non negative eigenvalues and is shown as,

$$\widehat{(-\Delta_N u)} = \mathbf{C}_{N,N}(u) \times (\omega_{x,N}^2 + \omega_{y,N}^2), \tag{3.24}$$

where $\omega_{x,N}^2$ and $\omega_{y,N}^2$ are two dimensional spectral discretization mesh.

Spatial Discretization of Edge Detector Function $g(u^n)$

The spatial edge detector $g(u^n)$ is formed by first achieving a fractional derivative with the spectrally discretized Laplacian $(\widehat{-\Delta_N u})$ shown in equation (3.24), then applying the inverse Fourier cosine transform and then finally achieving the reciprocal form with coefficient K. The fractional derivative is well defined and is achieved by exponentiating the spectrally discretized Laplacian to the $1-\varepsilon$ power [19]. Negation of the Laplacian is done before exponentiation because the Laplacian has negative eigenvalues and the fractional derivative must be applied to a quantity having positive eigenvalues.

$$(-\Delta_N u)^{1-\varepsilon} = \mathbf{C}_{N,N}^{-1}[\widehat{(-\Delta_N u)}^{1-\varepsilon}]. \tag{3.25}$$

The first step forms the fractional Laplacian in the spectral domain as signified by the (N,N) subscripts on the Fourier cosine transforms as follows:

$$\widehat{(-\Delta_N u)}^{1-\varepsilon} = [\mathbf{C}_{N,N}(u) \times (\omega_{x,N}^2 + \omega_{y,N}^2)^{1-\varepsilon}]. \tag{3.26}$$

The second step applies the inverse Fourier cosine transform to return the Laplacian to the spatial domain such that

$$(-\Delta_N u)^{1-\varepsilon} = \mathbf{C}_{N,N}^{-1}[\mathbf{C}_{N,N}(u) \times (\omega_{x,N}^2 + \omega_{y,N}^2)^{1-\varepsilon}]. \tag{3.27}$$

The final step involves squaring the result of equation 3.27, multiplying by k^2, adding one and inverting yield the spatial discretized edge detector $g(u^n)$ as shown below,

$$g(u^n) = \frac{1}{1 + k^2[\mathbf{C}_{N,N}^{-1}[\mathbf{C}_{N,N}(u^n) \times (\omega_{x,N}^2 + \omega_{y,N}^2)^{1-\varepsilon}]]^2}. \tag{3.28}$$

3.3.2 Discretized Solution u_{t+1}

The solution scheme forms the solution at each timestep form the product of the Fourier coefficients and exponential of the eigenvalue matrix times the timestep.

$$u^{n+1}(\omega_1, \omega_2) = \mathbf{C}_{N,N}^{-1}[e^{-\tau_{2a}\Delta t}]_{1,1}\mathbf{C}_{N,N}u^n(\omega_1, \omega_2)], \tag{3.29}$$

where u^n is the image matrix from the previous timestep. This scheme is solved using the **KSSFast** timestepping as described in sections prior to this point. A summary follows.

3.3.3 Summary of KSSFast timestepping

The **KSSFast** implementation is performed each times step through the four steps listed below.

- The Fourier cosine transform of the image matrix produces the Fourier coefficients as described in section 3.2.1.
  $\mathbf{C}_{N,N}(u^n) = \hat{u}(x,y,t_n) = \Sigma_{\omega_1,\omega_2=1}^{\infty}\hat{u}(\omega_1,\omega_2,t_n)\cos(\omega_1 x)\cos(\omega_2 y)$

- Calculation of the exponential form is done using the divided difference formula described in Section 3.2.3, for Fourier coefficients in the solution as,
  $[f(\tau_{2a}(\omega_1,\omega_2))]_{1,1} = f(\lambda_1) + \frac{f(\lambda_2)-f(\lambda_1)}{\lambda_2-\lambda_1} \times (A11 - \lambda_1)$,
  where λ_1 and λ_2 denote the eigenvalues, $A11 = \alpha_1$, $f(\lambda) = e^{-\lambda\Delta t}$, and the $(1,1)$ signifies that only the eigenvalues of τ_{2a} are used in the exponential, for frequency dependent terms.

- The approximate matrix-function vector product is formed by multiplying the matrix exponential by the each of the Fourier coefficients forming the spectral solution in Section 3.32.
  $\hat{u}^{n+1}(\omega_1,\omega_2) \tilde{=} [f(\tau_{2a})(\omega_1,\omega_2)]_{1,1} \times \hat{u}(\omega_1,\omega_2,t_n)$

- The inverse Fourier cosine transform is applied to return the spectral solution into the spatial domain.
  $u(x,y,t_{n+1}) \tilde{=} \mathbf{C}_{N,N}^{-1}[[f(\lambda)(\omega_1,\omega_2)]_{1,1} \times \hat{u}(\omega_1,\omega_2,t_n)]$

This completes the explanation of the **KSSFast** method.

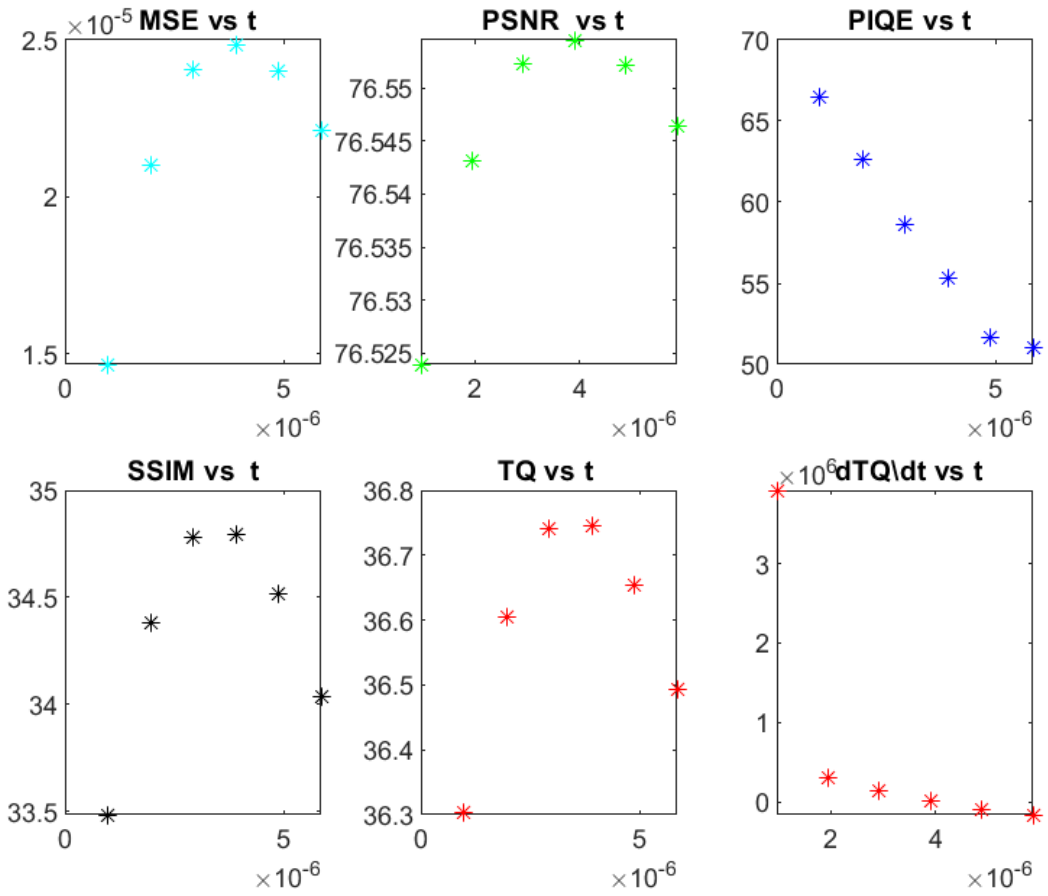

Figure 3.1: Total Quality vs Time in seconds, shown in top row: are MSE cyan, PSNR green, Piqe blue, and in bottom row: SSIM black, TQ red and dTQ/dt red for each timestep in one *dt* optimization loop

3.4 Adaptive Parameter Refinement

One of the impediments to automated and efficient image processing is the task of choosing parameter values that are suited for the particular image to be processed that are needed for the particular method used. Images vary in size, noise level, color and other structural qualities. Initially lacking insight as to what parameter values are best creates a tedious task, usually having a "trial and error" approach to determine good values for the parameters required. This task typically requires a great amount of human intervention. It is our goal to be automate this process and remove any need for human intervention. This requires an algorithm with an intelligent approach to setting the initial values for each parameter, as well as the range the values need to span as the refinement task proceeds. The proposed method bases initial parameter settings on the initial quality measures of the noisy image at

the local sub-block level. Later the algorithm determines how to revise the parameters based on how the denoising process advances towards improved quality.

The proposed method uses three parameters: timestep dt, edge threshold k, and regularization parameter ε. The method aims to maximize efficiency by starting with the largest values for each of these three parameters as defined by the end of an testing range for each. The denoising process commences with these maximal starting values, and each parameter is individually reduced and optimized as the denoising process is repeated with successively improved results. This is achieved with three nested loops, implemented from the inside out, such that each loop optimizes each parameter individually. The innermost dt optimization is done first, then the middle loop optimizes k, and finally the outermost loop optimizes ε. The main optimization mechanism is based on improved quality after each **KSSfast** application in the timestepping t loop which is located within the inner dt loop. By comparing successive quality states after each timestep the optimization mechanism finds the maximization of total image quality defined by the Total Quality measure TQ, a overall quality score which is a scaled sum of individual quality measures MSE, $PSNR$ and $SSIM$ as described in Section 1.3.5.

3.4.1 Parameter Optimization

In the innermost t loop of the algorithm, the parameter dt is set to the starting value, the **KSSfast** timestepping solver is run using and initial t and this dt value. At each timestep the quality TQ is measured for the denoised image. This is repeated at successive t values advanced by dt at every subsequent iteration, until the TQ value is found to increase to a maximum as $TQbest$. Termination occurs when a $TQbest$ is found within a specified tolerance, or the final time T is reached, usually the former. This takes a minimum of 5 timesteps as illustrated in Figure 3.1. Figure 3.1 shows each raw quality measure as well as TQ for each timestep as a particular sub-block being denoised. The figure contains subplots for MSE, $PSNR$, $PIQE$ in the top row and $SSIM$, as well as the combined Total Quality TQ and its change per timestep dTQ/dt in the bottom row at each timestep along the horizontal axis. For this particular image, three increasing steps results in the maximumn quality state, followed by two decreasing steps , hence it takes a minimum of 5 timesteps to find a $TQbest$ value. Note the $PIQE$ measure was removed from the TQ calculation, because had oscillations that triggered false values in the optimization algorithm. for the each separate quality measures *mse*

Then the next outer k loop is initiated to optimize the k parameter. The same process is repeated to optimize k, beginning with the optimum dt from the inner loop, but now reducing

the k parameter is successively by a k-step variable. The $TQbest$ at the smaller k value is compared with the $lastTQbest$ at the prior larger k value. As with the dt optimization previously described, this k optimization will terminate when the $TQbest$ values are no longer increasing, confirming the optimum k value and the loop terminates.

Finally the outermost ε loop begins, repeating the same process as is done as previously described for the dt and k optimizations, but using successively reducing values of ε through a eps-step value to determine successive $TQbest$ values terminating when no improvement in $TQbest$ occurs. This simple approach to determine a maximum will effectively achieve optimum parameters if the parameter values' range contains the optimum quality state. This range is set by the initial values for each parameter: global-dt, global-k and global-epsilon and by the corresponding reduction factors: $\frac{1}{2}$ for dt , and k-step and eps-step for k and ε respectively. To tune the model, adjustments were made to the reduction factors as well as a parameter tolerance values for each parameter used to determine if each parameter optimization achieves a maximumn $TQbest$. Adjusting these tuning parameters allows customization of the parameter optimization to a particular image type was done during the initial development. For all experiments in this research, these tuning parameters are held constant, so that the optimization algorithm is consistently applied to all test images. The constant tuning parameters include: initial values, reduction factors, tolerances used for each parameter in the the parameter optimization. Tables 4.3 to 4.10 show the final optimized parameters for each test.

The objective function for Total Quality is a function of multiple variables, including the three parameters to be determined, but also image based variables that may not all be known. There is not a defined function for Total Quality, and so a derivative free approach to optimization is used, similar to the Nelder-Mead method, but does optimization for a single parameter at a time, which simplifies the method. The method is applying basic optimization principles in which a desired value is sought by successive steps after which a comparison is done to determine if the function is increasing. Feedback if it is not increasing is used to modify parameters to maintain a path to increasing quality [50]. This continues until the objective function reaches a state in which it no longer changes, within some defined tolerance, or begins to decrease.

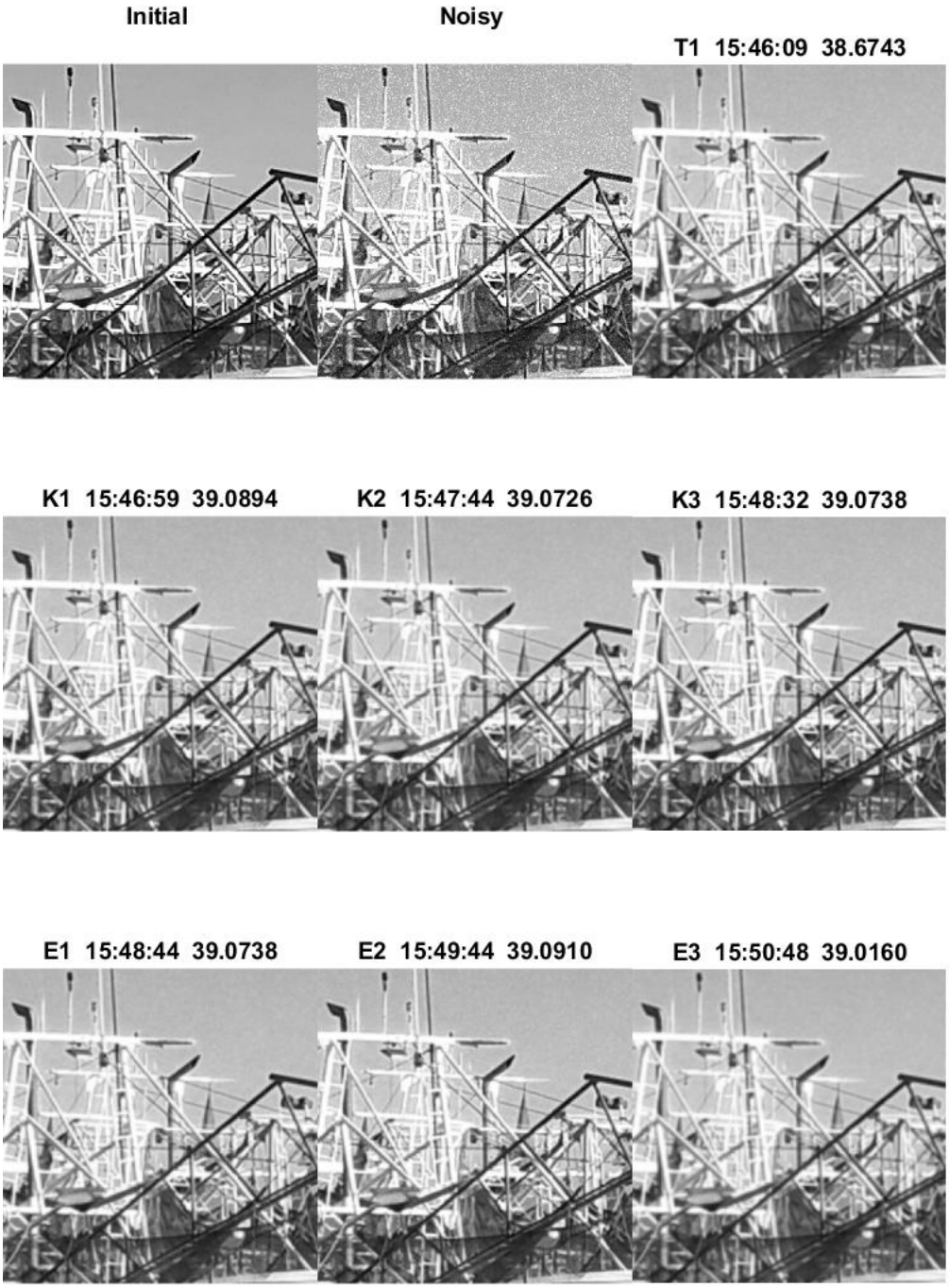

Figure 3.2: Parameter Optimization Progressive View For Red Block 11 of Boat Image with $n = 0.010$ noise: Initial, Noisy and dt (top left to right), k (middle left to right and ε (bottom left to right)

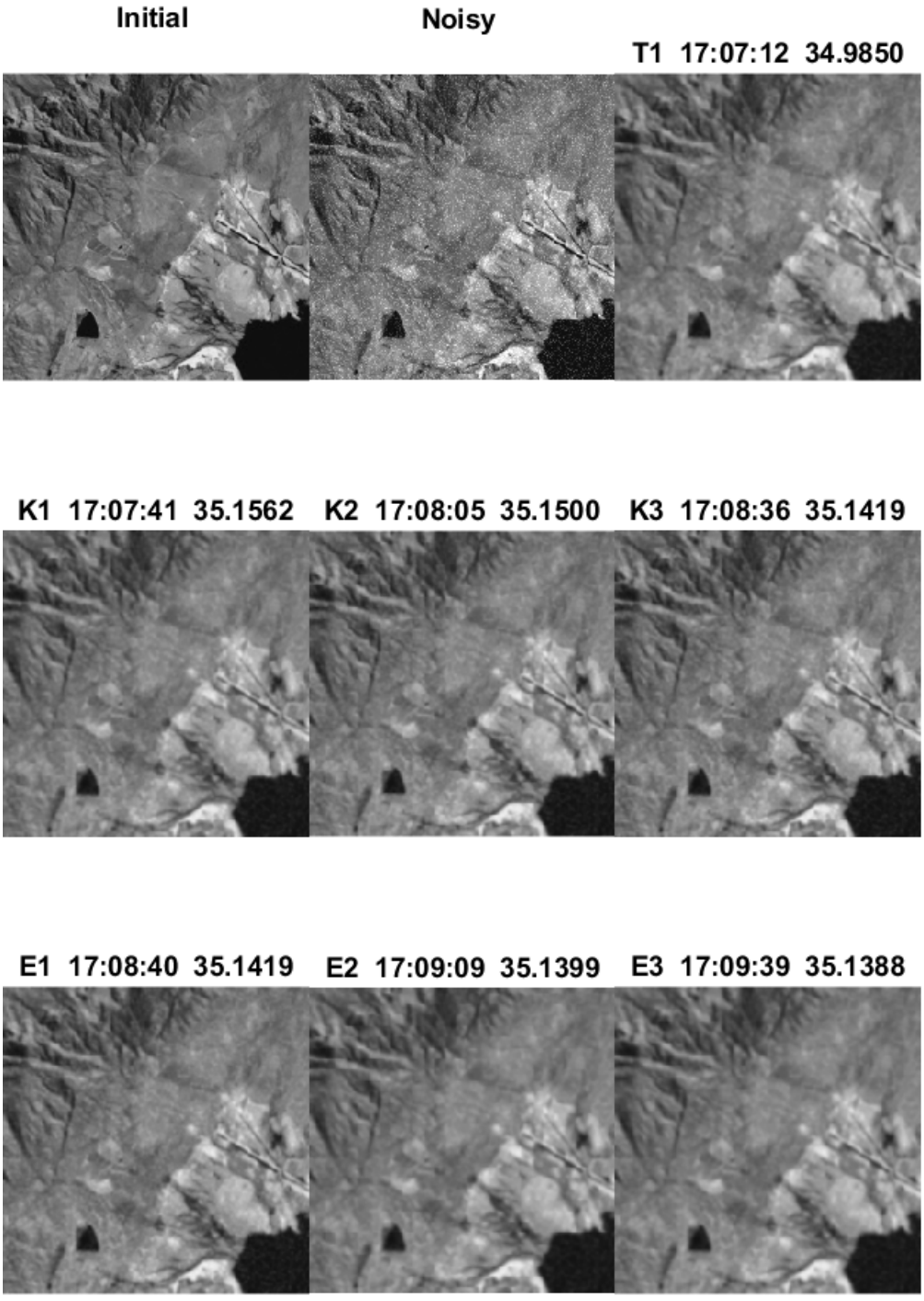

Figure 3.3: Parameter Optimization Progressive View For Red Block 11 of Landsat Image with $n = 0.010$ noise: Initial, Noisy and dt (top left to right), k (middle left to right and ε (bottom left to right)

Optimizing Total Quality is comparable to finding the maximum on a function that is always concave down yet unknown. The general shape is known but not a specific formula and so a quick determination of a vertex maximum or derivative based minimization is not possible. The Total Quality measure TQ is a positive sum of three quality measures $MSE*$, $PSNR$ and $SSIM*$, arranged so that it is always positive as described in Section 1.3. Each of these measures are positive quadratics as noise is decreased, thus so is the objective function. This simple method is expected to be robust and to work with a wide range of test images, to efficiently control the denoising process with no human intervention once the initial starting parameter values, stepping and tolerance settings are input.

3.4.2 Finding Maximum Quality

It has been observed through initial testing of the optimization algorithm that the following relationships of the parameters dt, k and eps need to be set so that an optimum can be determined. By starting with a large dt and reducing it, the optimum quality can be found in a minimum of attempts. The k loop is also started with a large initial value that is reduced, to ensure that the maximum $TQbest$ can be found. The ε value starts large and is reduced, to optimize the quality. The value of ε has the effect widening the TQ curve, allowing for small refinement of the TQ maximum.

The Initial stage of processing is to set the paramter values using the parameter optimization routine for the first block, the Red block row 1 column 1 of the divided whole image. In which ideal values for dt, k and ε which are specifically tuned to that image. These values are then used as starting values for all the remaining blocks. Following denoising is performed on each sub block and for each color. The whole image and block R11 denoising results are shown in the figures that follow. Figure 3.2 shows this optimization process for the Red $1,1$ block of the Boats image. The top row shows the initial and noisy images and the final dt optimized state labled $E1$ in the top row. The second row shows the three k optimized states labled $K1$-$k3$, and the last row shows three ε optimized states labled $E1$-$E2$. The runtime and $TQbest$ values are shown for each subplot, and it should be noted that although subsequent $TQbest$ values may not always increase, they are within the tolerance for that level of optimization.

Because all parameters begin large, termination is flexible to meet the needs of the image being processed. For any image to which the algorithm is applied, a point will be found at which the image will reaches an optimum quality state, and the routine will stop there. This ensures that overworking that results in excessive blur, degrading the image quality will not occur. As soon as TQ stops increasing, termination will happen. The optimum

$TQbest$ will vary from image to image but does exist. Though proper parameter ranges and large initial values the optimum $TQbest$ will be found. Controlling termination will prevent issues related to overworking and backward diffusion instabilities that create excessive blur and artifacts such as stair-casing.

3.4.3 Block Processing

Block procsssing begins with trimming and cropping images to be square. Then depending upon the size of the image, it will be further subdivided into square sub blocks. This readily allows for parallel processing as well as more localized treatment of the image for denoising. This also has some efficiency benefits as it takes less time to process smaller blocks, and the parameter optimization required can be minimized by sharing the optimized parameters for initial settings, and recycled for final run values, among blocks in the same image that share similar characteristics that are reflected in similar local block level initial quality scores.

Parameter Optimization Time: The parameter refinement algorithm requires significant time to process an image as it involves at minimum two dt, k and ε optimizations. Each t optimization typically takes about 3 seconds, so each parameter optimization for dt, k and ε takes at minimum 15 seconds for the first and confirming set of five t loop iterations Typically this results in a minimum of 180 seconds, or 3 minutes for a full parameter optimization of a single optimized block. Most images have 4 blocks in three colors so optimizing every block would typically take 90 minutes for the whole image. Table 4.21 shows the parameter optimizaton times for each image tests. These time measures will vary according to the hardware used to run the algorithm. This research used a computer with an Intel Core $i7-8550$ CPU 1.8GHz.

Sharing Optimized Parameters: Typically within a given image that is color balanced, the sub blocks into which it is divided will have some degree of similarity. A color balanced image has sufficiently similar color intensity ranges for each color matrix. Through early testing it was observed that this is beneficial to denoising success based on parameter sharing. In images containing a particular color matrix with much lower pixel intensity ranges, than the other color image, the denoising was observed to be much less successful. It is hypothesized that this is because the added noise pixel intensities far exceed those in original image data for that color image. For this reason it is sensible to consider reusing the optimized parameters from the first block of an image for sub blocks that are similar enough. Initial testing of the parameter optimization algorithm on each sub block of each color image showed no difference in the parameters optimized between the sub blocks.

Reduced Processing time:

The initial parameter optimization testing showed that optimization need only be done for the first sub block processed, a considerable savings results as this reduces the parameter optimization time from 90 minutes to 9 minutes, for an overall runtime of 20 minutes. Refer to Table 4.21 which shows Parameter optimization values and times for the Red $1,1$ block of each image.

Chapter 4

Results

Experiments are performed to denoise images with a blocking scheme and parameter optimization to suit each image. Preprocessing each image includes cropping to a square size, then dividing into smaller, single color sub blocks to which identical method noise is added. To ensure testing of the parameter optimization algorithm is uniform and unbiased, all experiments begin with the same initial state. This state is defined by parameter values for dt, k and ε, algorithm tuning settings of parameter step size, maximum number of iterations and loop termination quality tolerances. Experiment results for all test images consist of figures for the initial noisy and final denoised states of images as well as corresponding tabular output of quality, time and parameter settings which are included in the sections that follow.

4.1 Test Images

Experiments were performed on the test images shown in Figure 4.1, with noise variances 0.010 and 0.0044. These images represent a variety of subjects, contain a mix of colors and a mix of high detail and uniform regions. The test images are identified by abbreviations of the file name as follows: the USM Administration Building will be called "USM"; the Biloxi Shrimp Boats image will be referred to as "Boats"; the LandSat8 Satellite image of South Africa will be identified as "Landsat"; and the "Peppers" image needs no abbreviation. These images were obtained from various publicly available sources, except for Boats image, which this author captured using a cell phone camera [43,44,14].

Figure 4.1: Test Images include the Peppers (top left), USM (top right), Landsat (bottom left), Boats (bottom right)

4.2 Experiment Results

Results include figures and tables arranged into sections by each image and noise level for a total of eight tests. In this research parameter optimization is performed on only the first Red block (R11) for each image, denoising is performed on each of the 12 blocks in each image, and then the twelve denoised blocks are recombined into a three whole color images on which quality measurements quantify the denoising performance of **KSSFast** and the six other denoising methods for comparison. The figures showing initial, noisy and denoised images will appear first in this chapter and will be in three sizes: whole image,

single block or cropped size. The numerical tabular data will follow appear toward the end of the chapter and will include parameter optimization for each first Red block, and quality and time information will be shown for each test. The final pages of this chapter offer summaries of all tests. Table 4.1 contains some abbreviations as used to describe the images in Figures in the sections that follow.

Table 4.1: Summary of Abbreviations Used in Test Image Results

Abbreviation	Description
Blk	Image Sub block such as: the first Red is R11 and last Red is R22
MTHs	Soft Thresholding Denoising Comparative Method
MTHh	Hard Thresholding Denoising Comparative Method
MMV	Moving Median Denoising Comparative Method
MW	Weiner Filter Denoising Comparative Method
MS	Convolutional Smoothing Denoising Comparative Method
IDF	Diffusion Denoising Comparative Method
KSS	KSSFast With Parameter Optimization

4.2.1 Landsat Test Results Noise 0.010

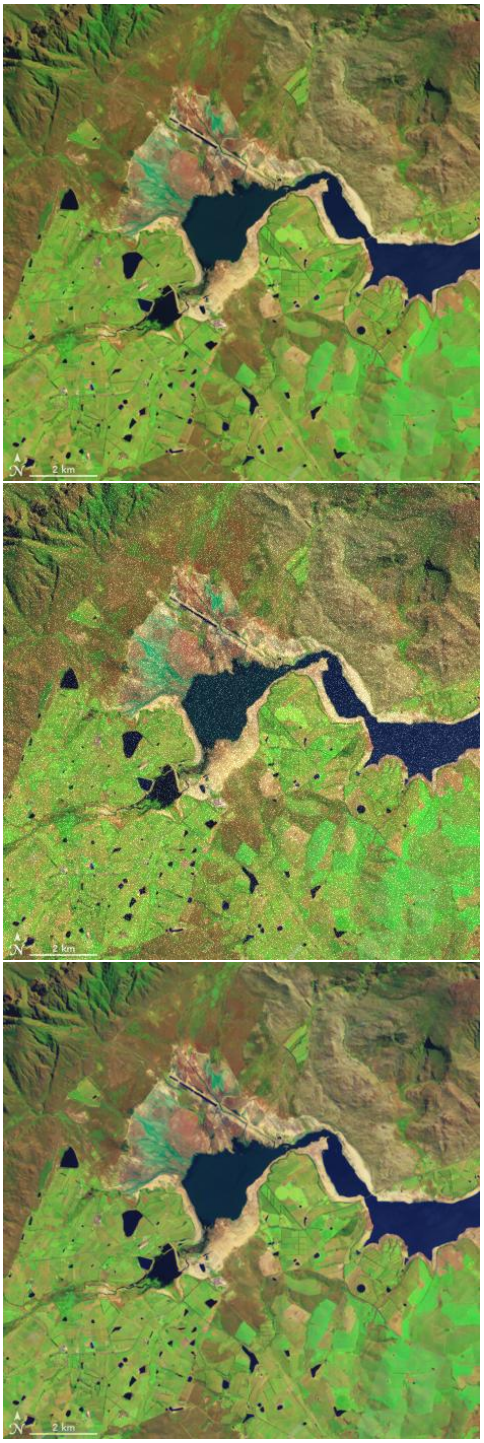

Figure 4.2: Landsat RGB Denoised Whole Images Initial (top), Noisy with Noise 0.010 (middle), and KSS Denoised(bottom)

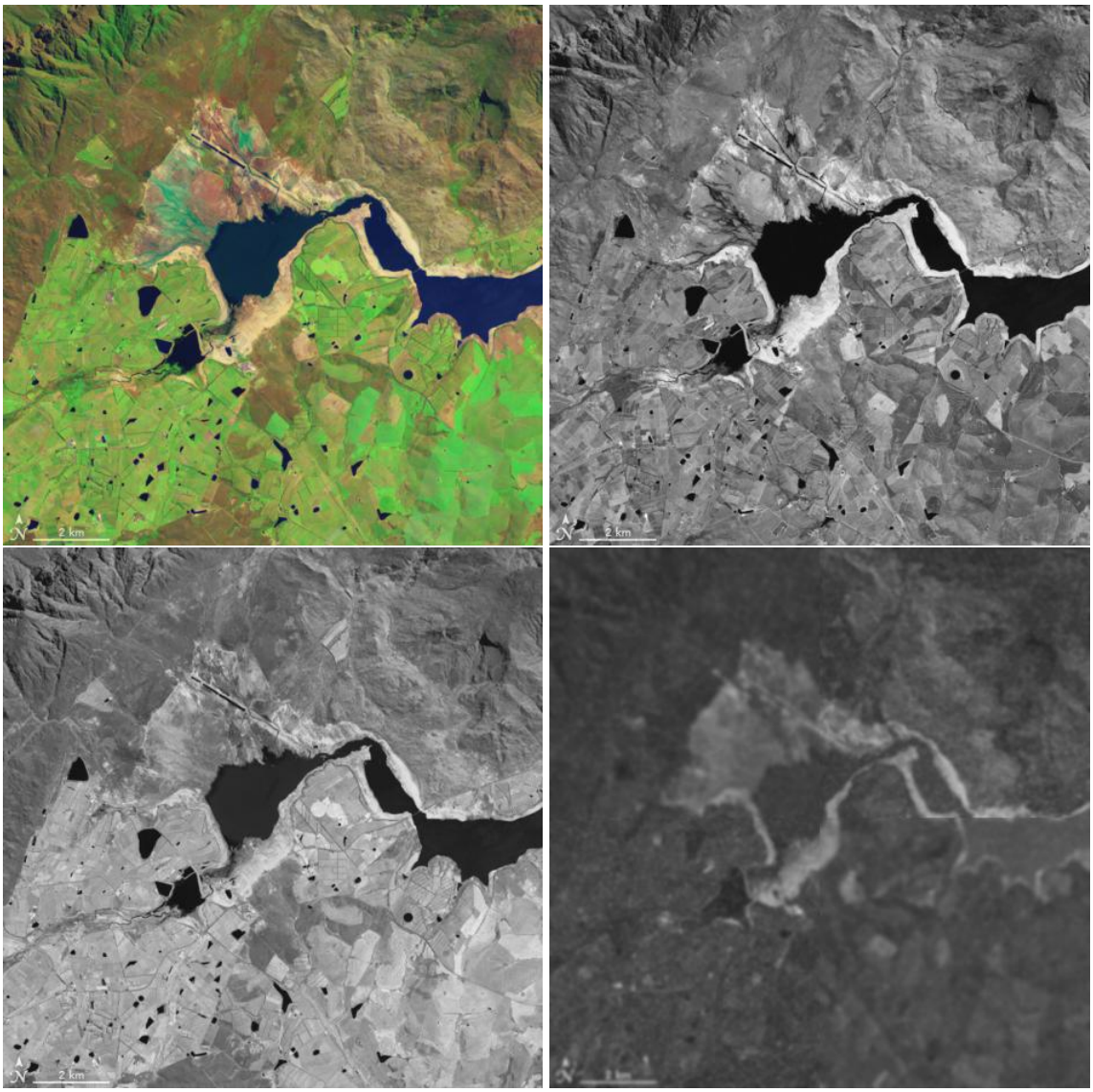

Figure 4.3: Landsat Whole Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.010 noise

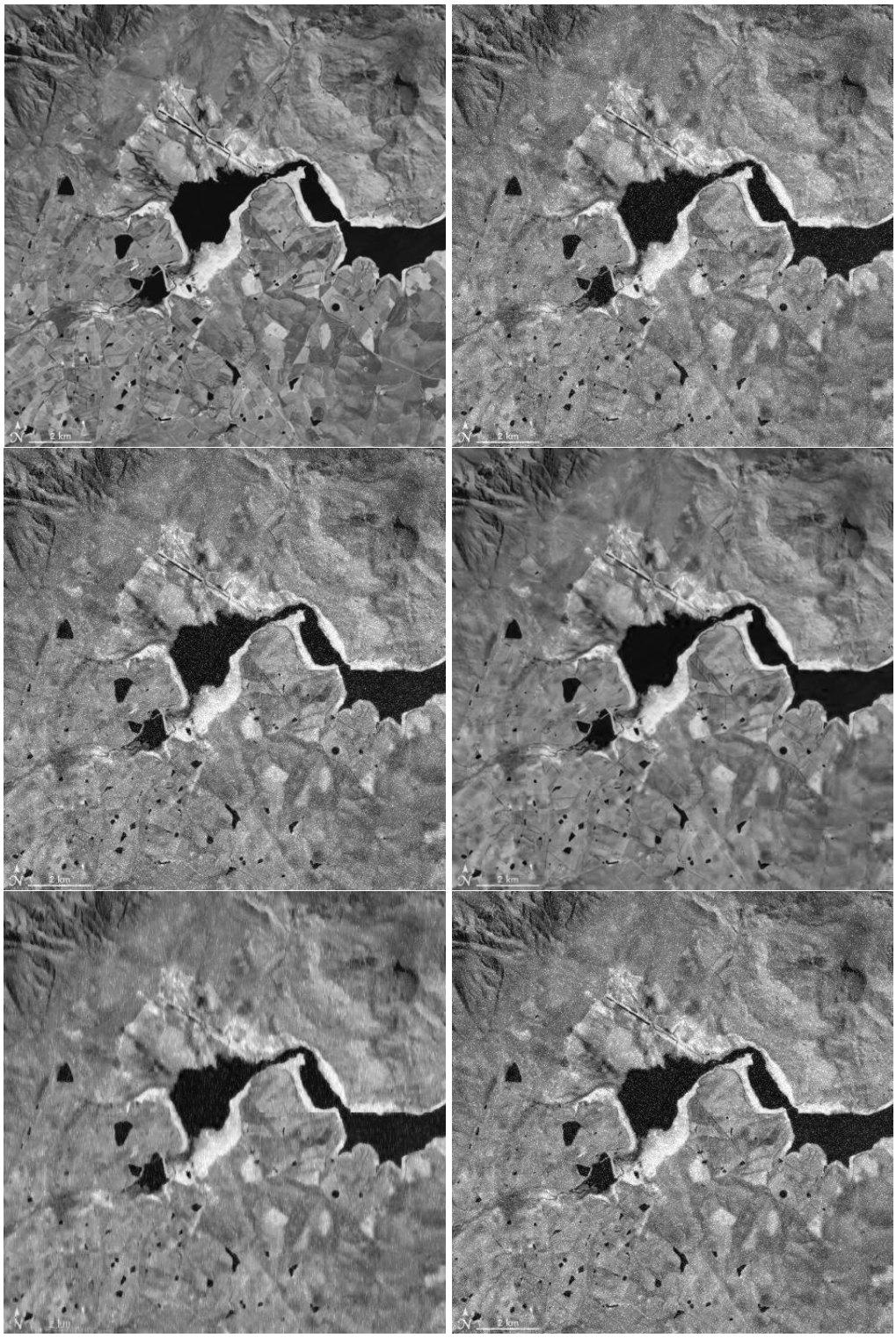

Figure 4.4: Landsat Red Whole Images: Initial and Noisy 0.010 (top left and right) and Denoised by IDF and KSS (middle left and right), and MMV and MV (bottom left and right)

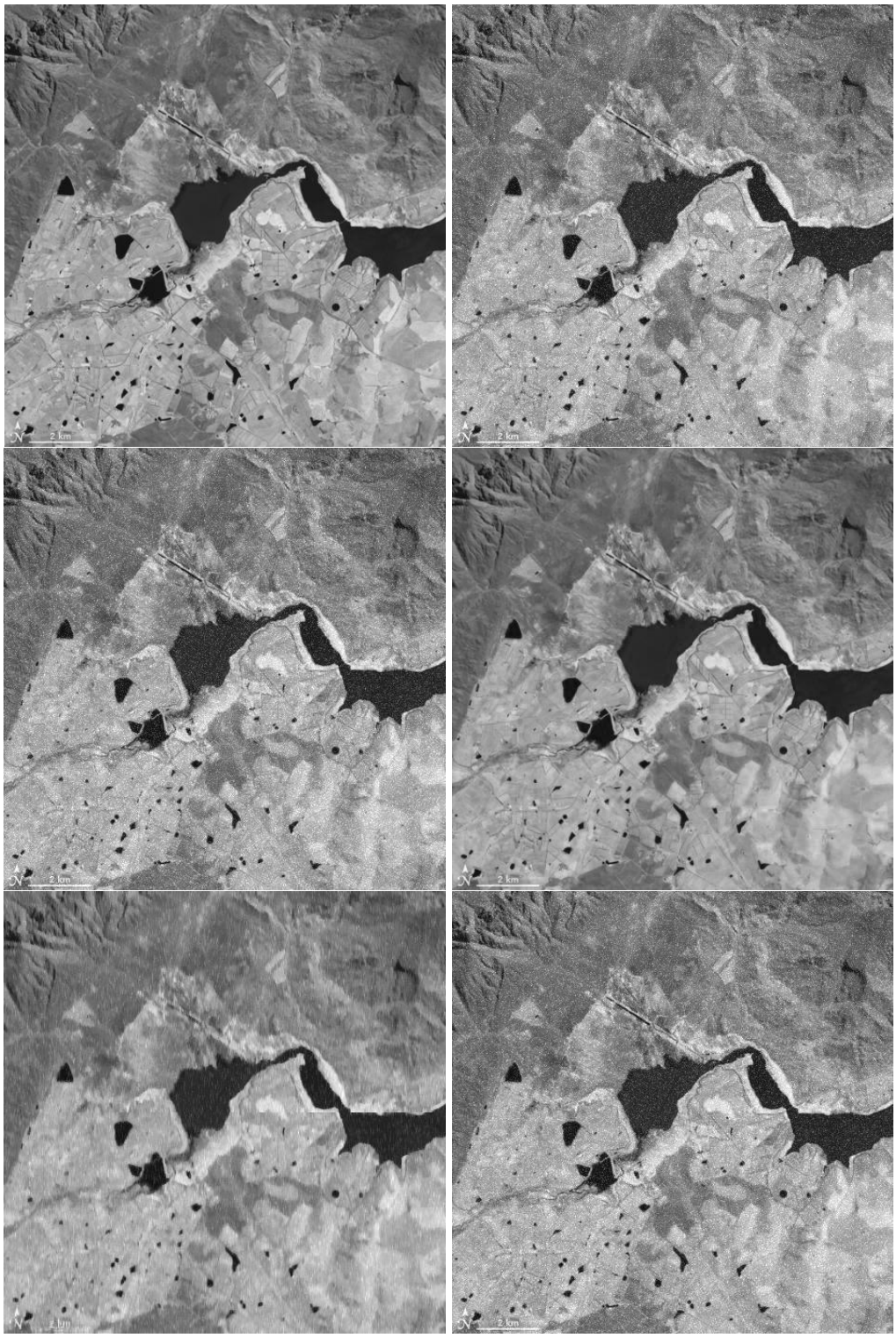

Figure 4.5: Landsat Green Whole Images: Initial and Noisy 0.010 (top left and right), and Denoised by IDF and KSS (middle left and right), and MMV and MV (bottom left and right)

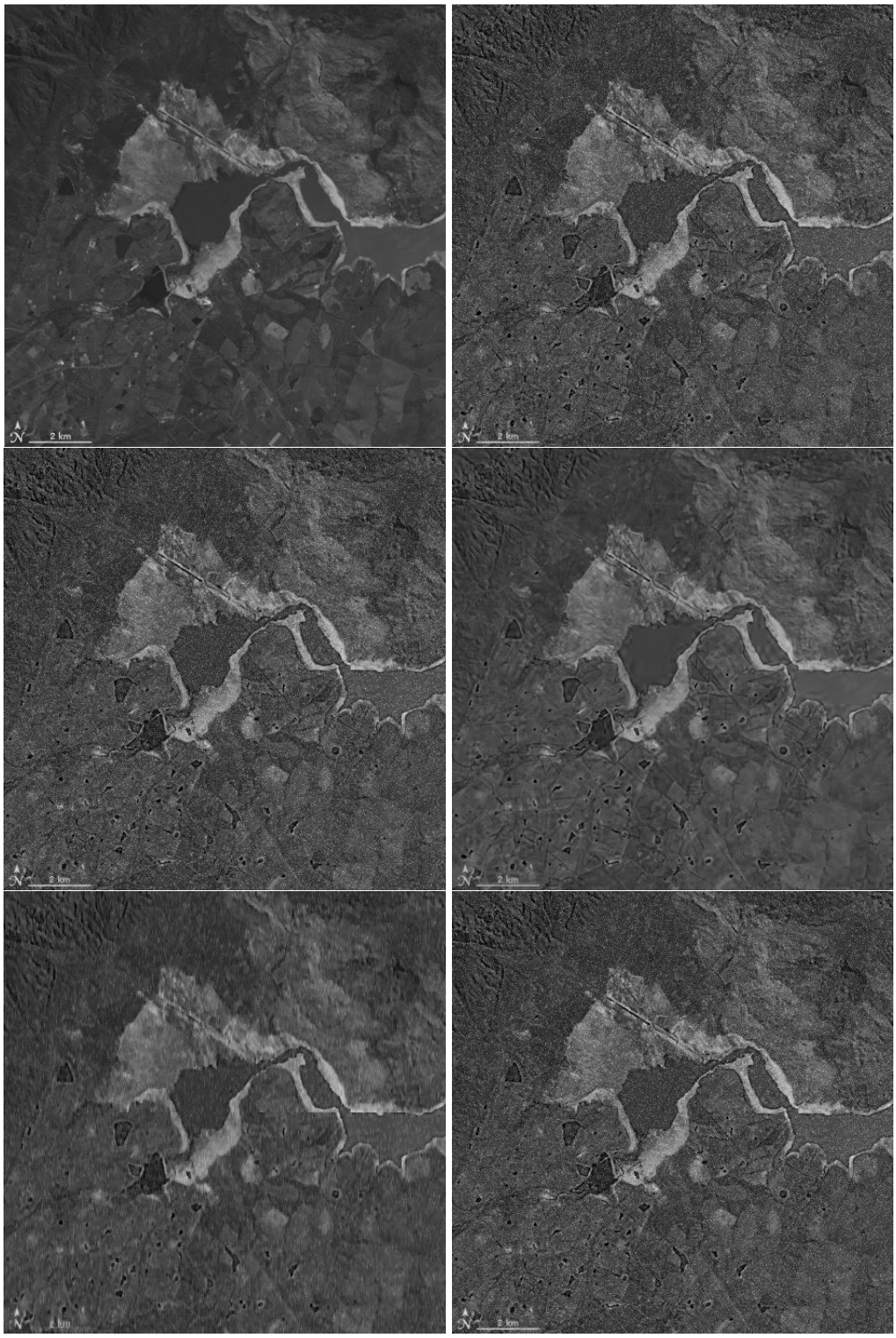

Figure 4.6: Landsat Blue Whole Initial and Noisy images (top left and right) and denoised images by IDF and KSS Methods (middle left and right) and MMV and MV methods (bottom left and right) with noise 0.010

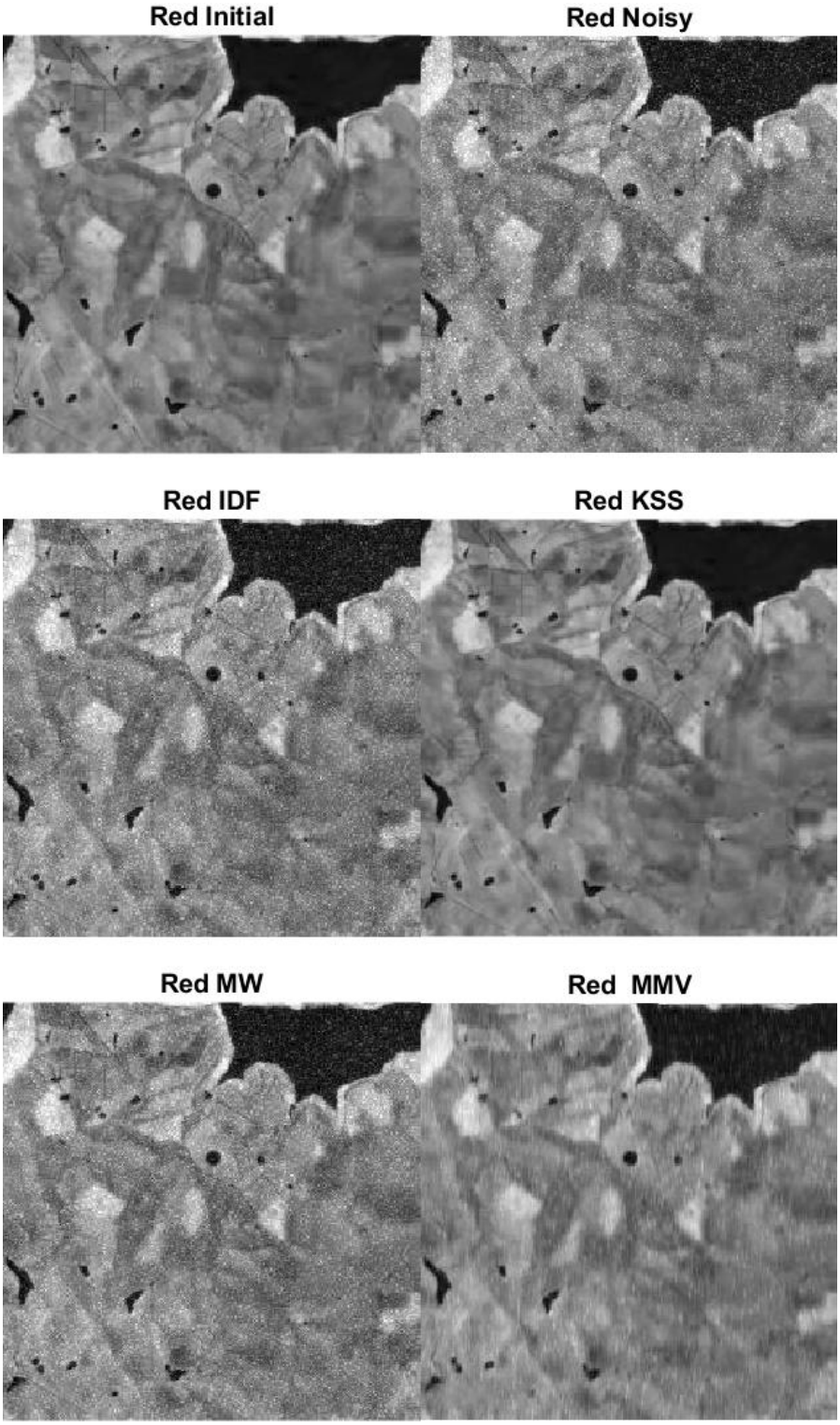

Figure 4.7: Landsat Red Block 22 Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

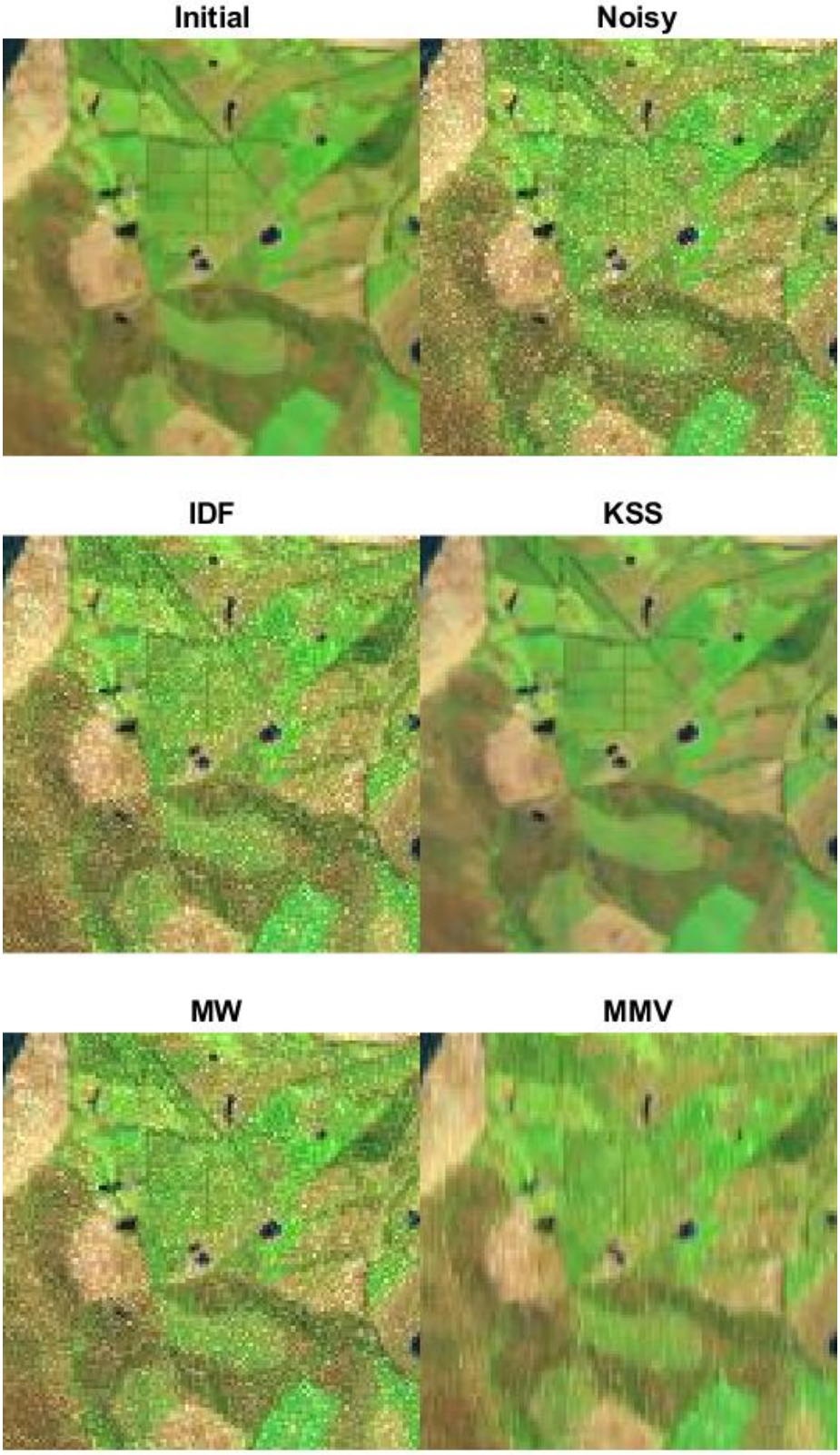

Figure 4.8: Landsat RGB Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

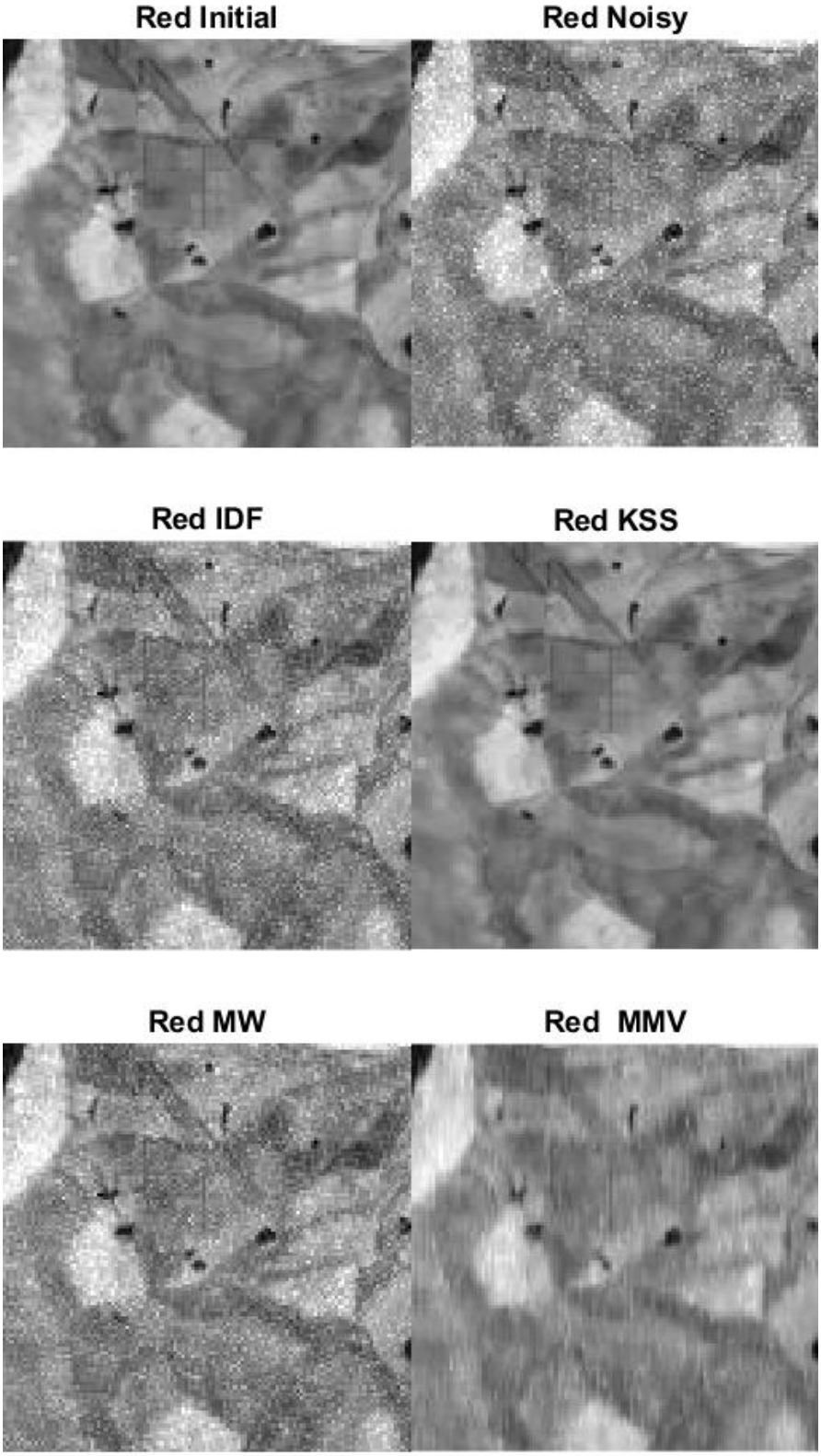

Figure 4.9: Landsat Red Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

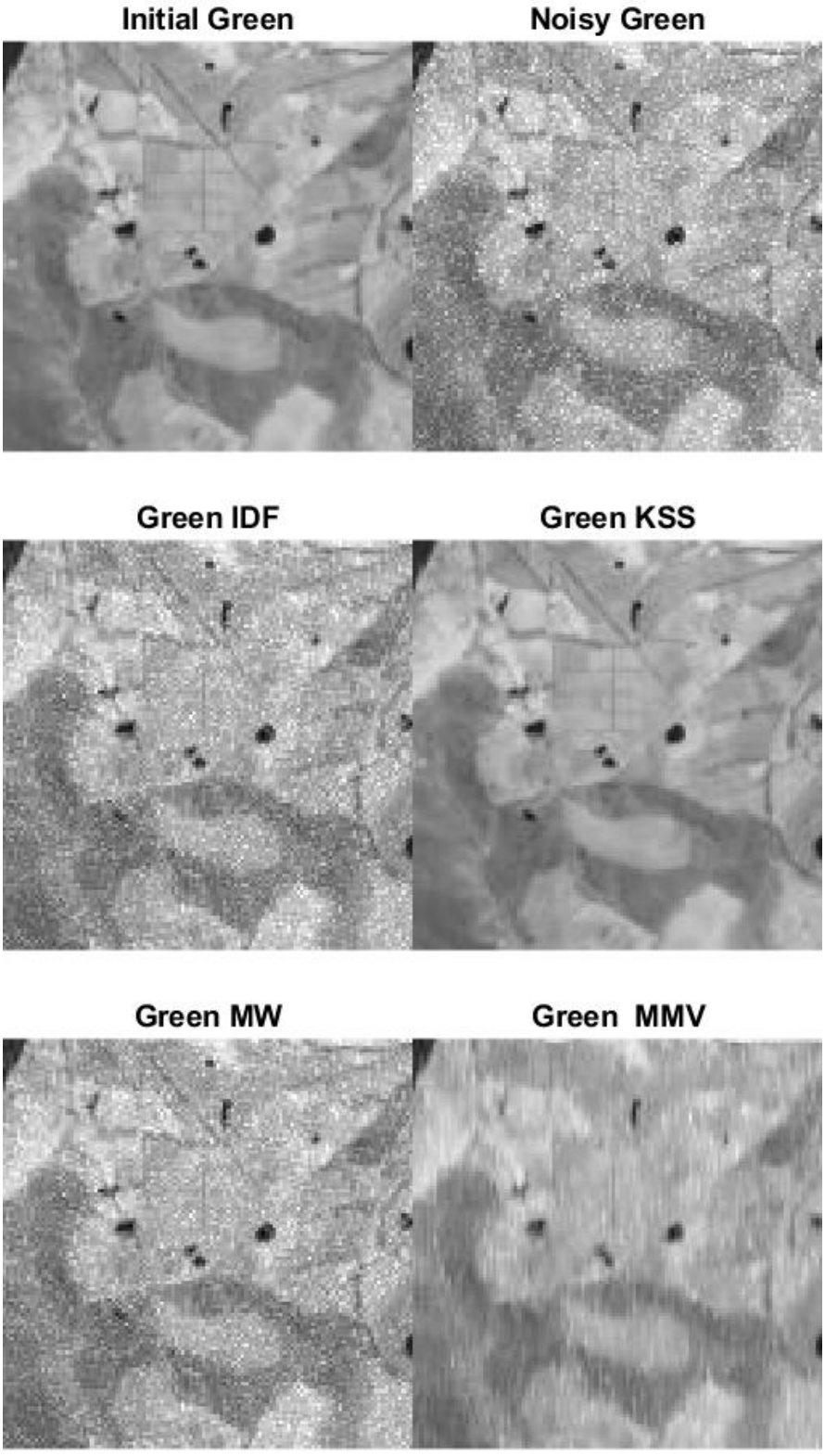

Figure 4.10: Landsat Green Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

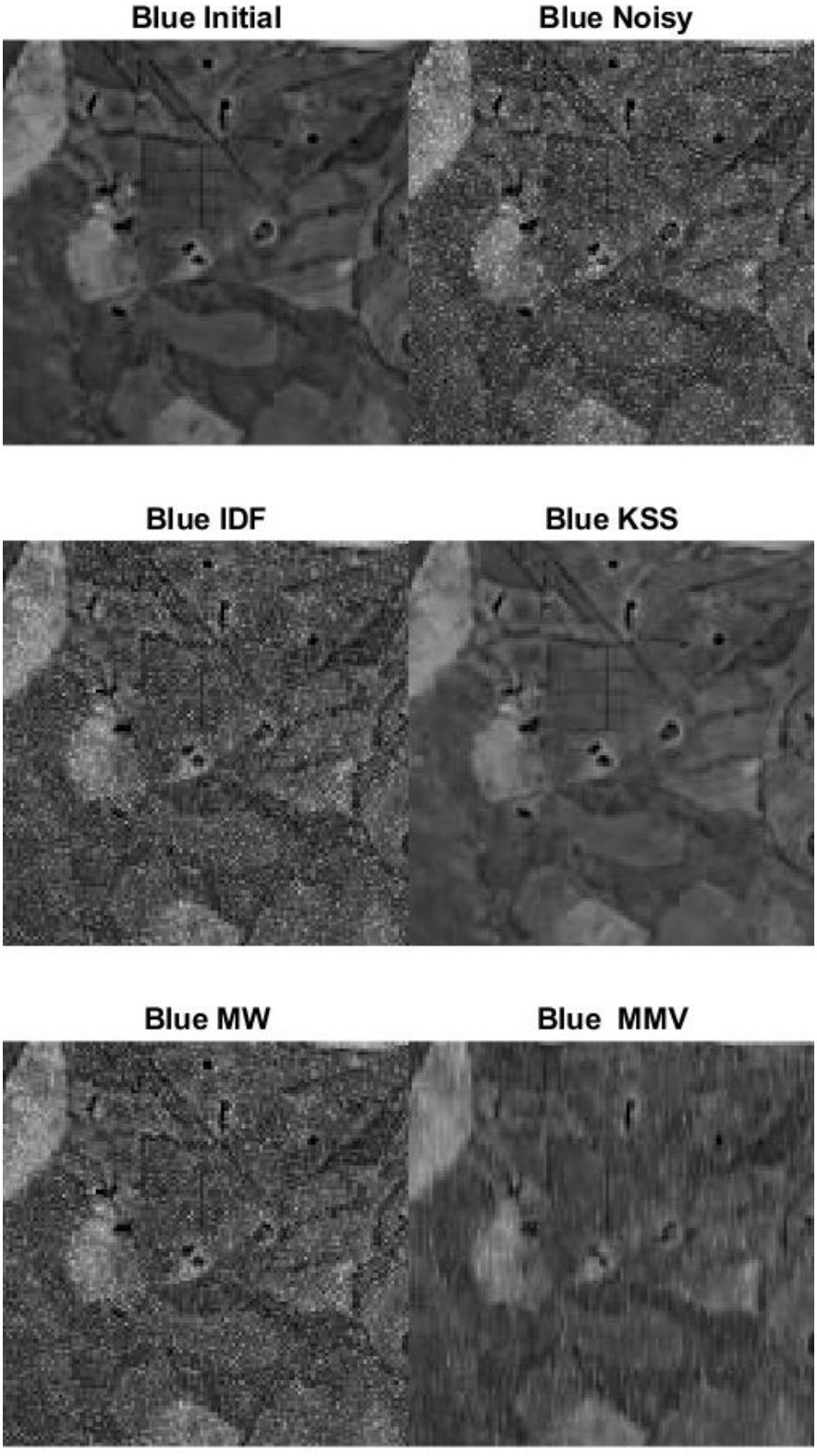

Figure 4.11: Landsat Blue Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

Figures in this section show the Landsat image in initial, noisy $n = 0.010$, and denoised states as whole images in Figures 4.3 to 4.7, block images in Figure 4.8 and cropped images in Figures 4.9 to 4.12. In all images the denoising is successful for the KSS method in comparison to all other methods. KSS has removed the noise and doesn't have added blur as the regularization parameter is able to sharpen the image. The comparison methods IDF, MW have lingering noise and graininess and the MMV has some blur.

4.2.2 Landsat Test Results Noise 0.0044

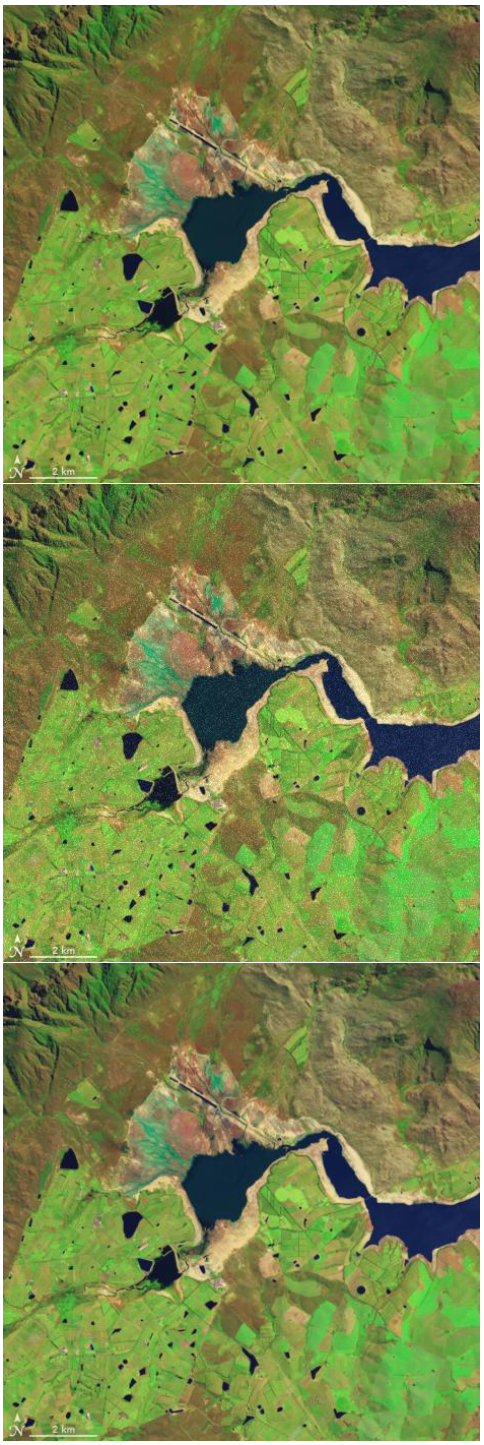

Figure 4.12: Landsat Whole Initial (top) Noisy $n = 0.0044$ (middle) and KSS Denoised (bottom) RGB images

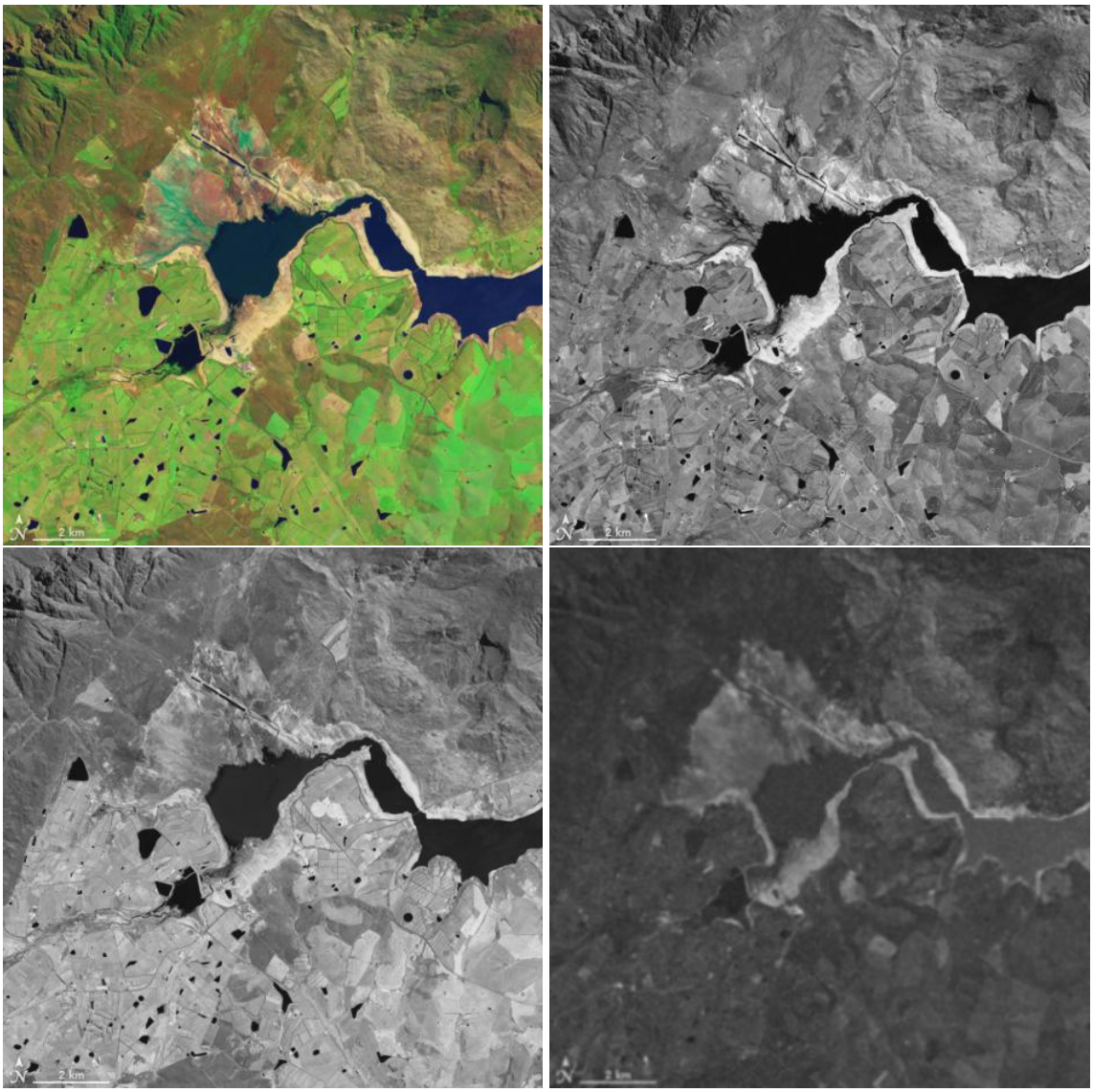

Figure 4.13: Landsat Whole Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.0044 noise

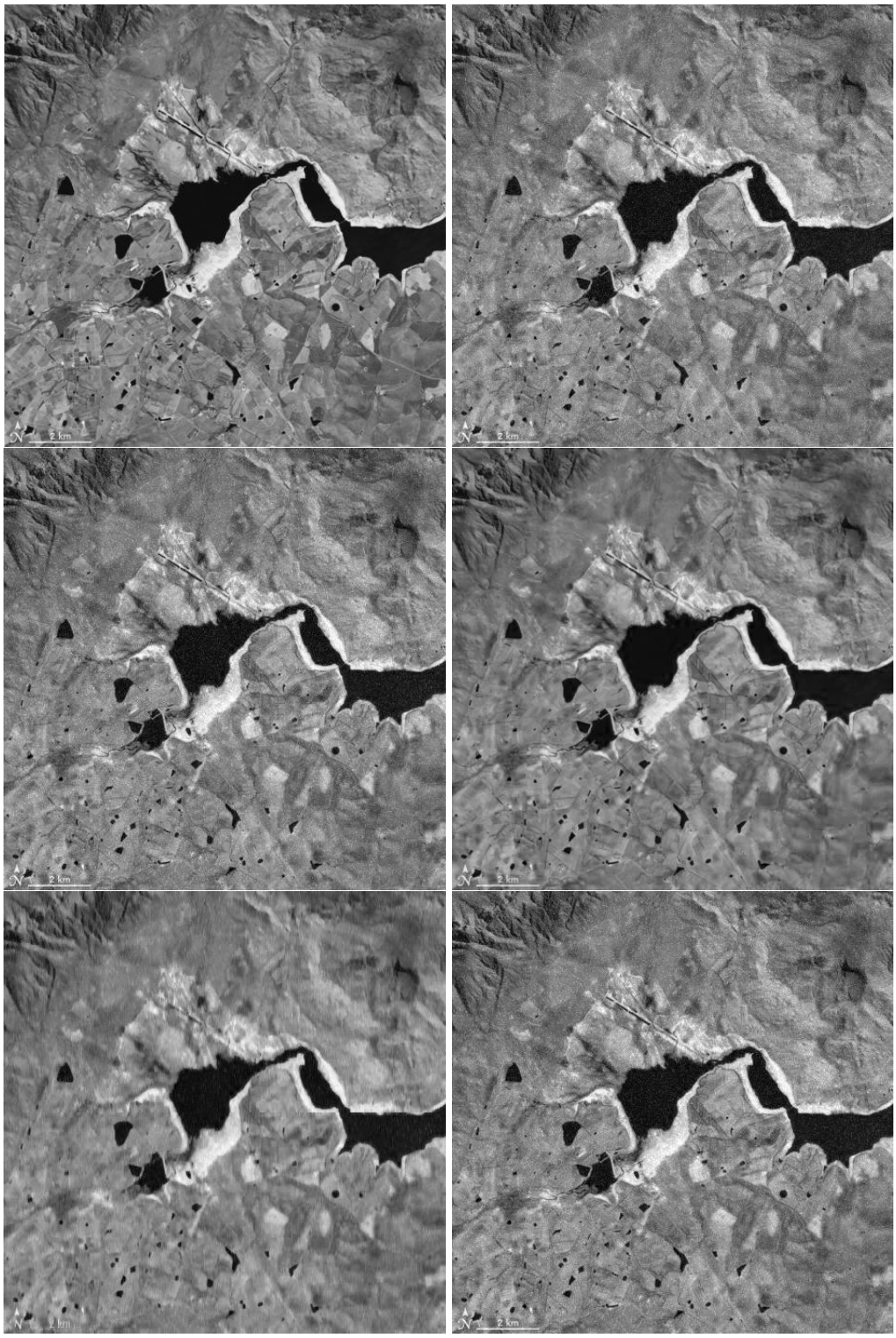

Figure 4.14: Landsat Red Whole Images: Initial and Noisy 0.0044 (top left and right) and Denoised by IDF and KSS (middle left and right) and MMV and MV (bottom left and right)

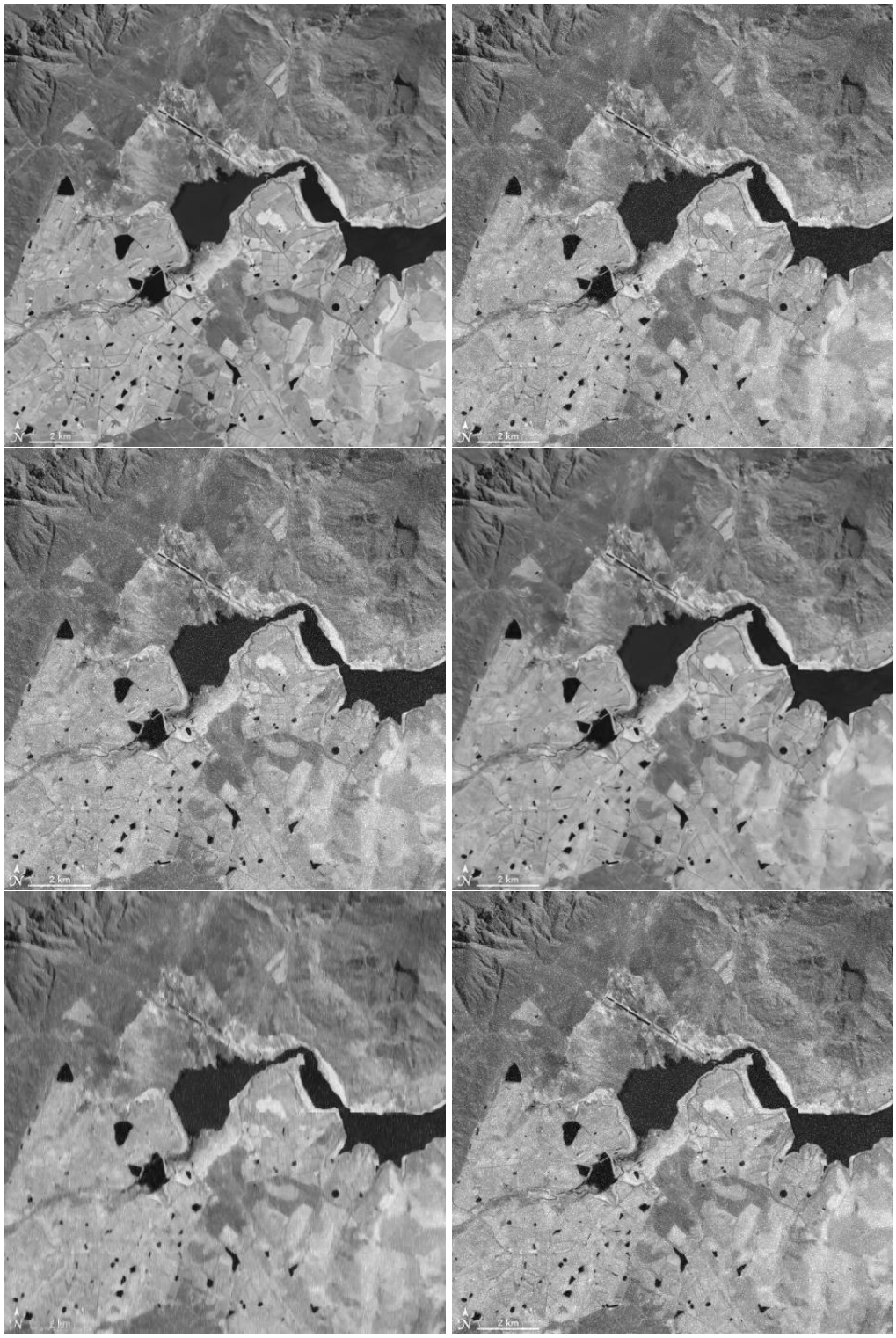

Figure 4.15: Landsat Green Whole Images: Initial and Noisy 0.0044 (top left and right), and Denoised by IDF and KSS (middle left and right) and MMV and MV (bottom left and right)

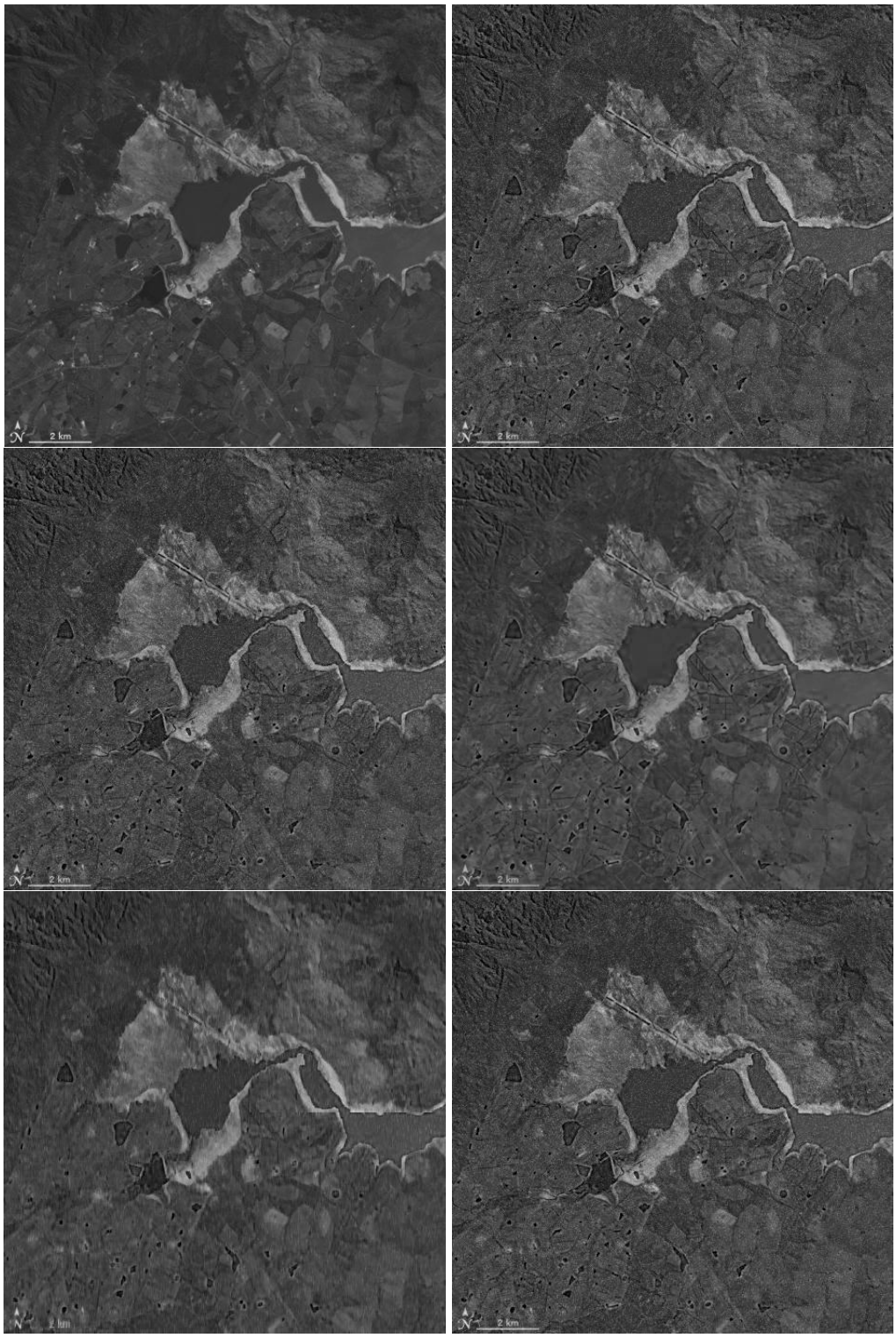

Figure 4.16: Landsat Blue Whole Images: Initial and Noisy 0.0044 (top left and right) and Denoised by IDF and KSS (middle left and right) and MMV and MW (bottom left and right)

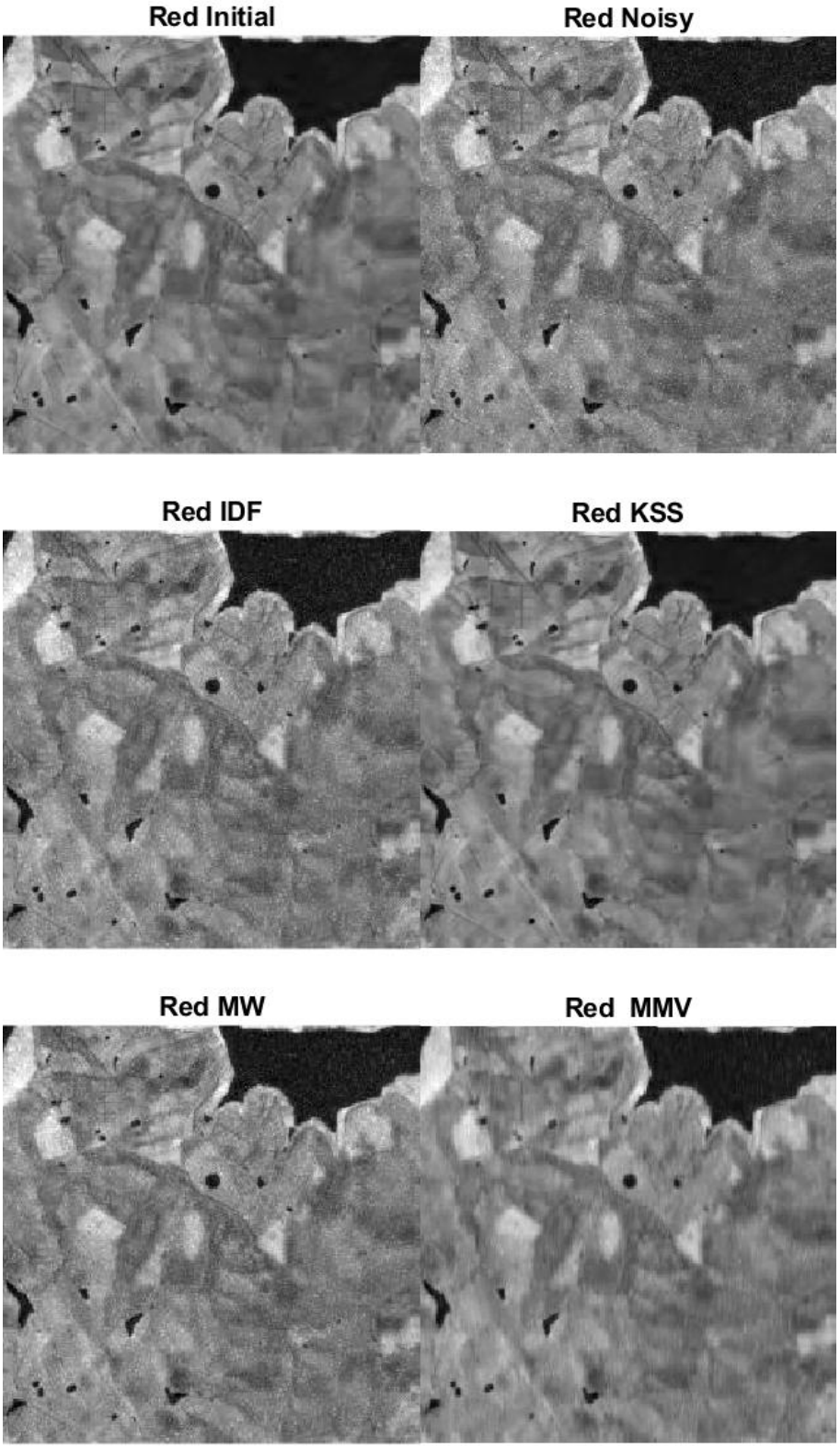

Figure 4.17: Landsat Red Block 22 Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

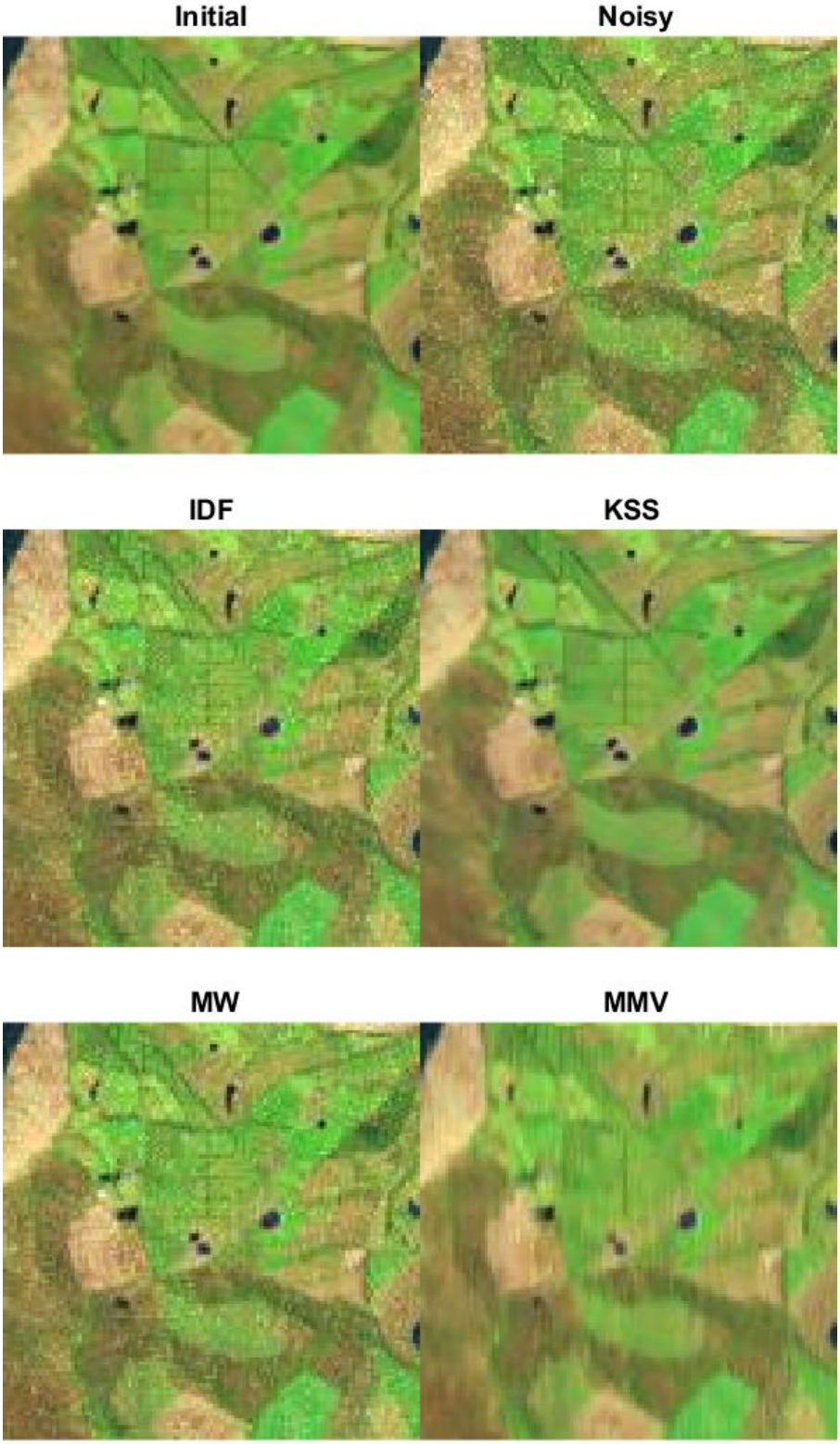

Figure 4.18: Landsat RGB Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

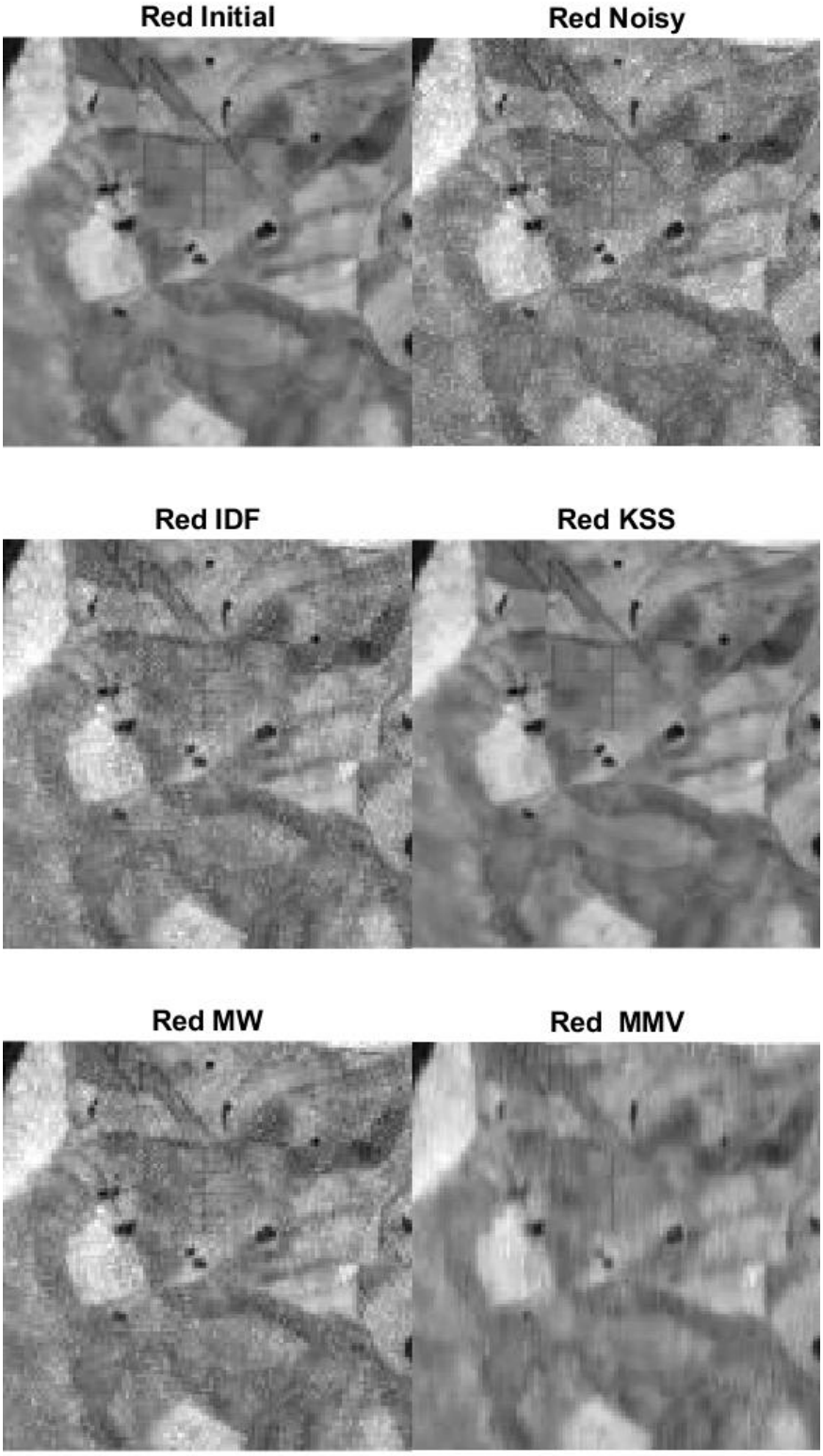

Figure 4.19: Landsat Red Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

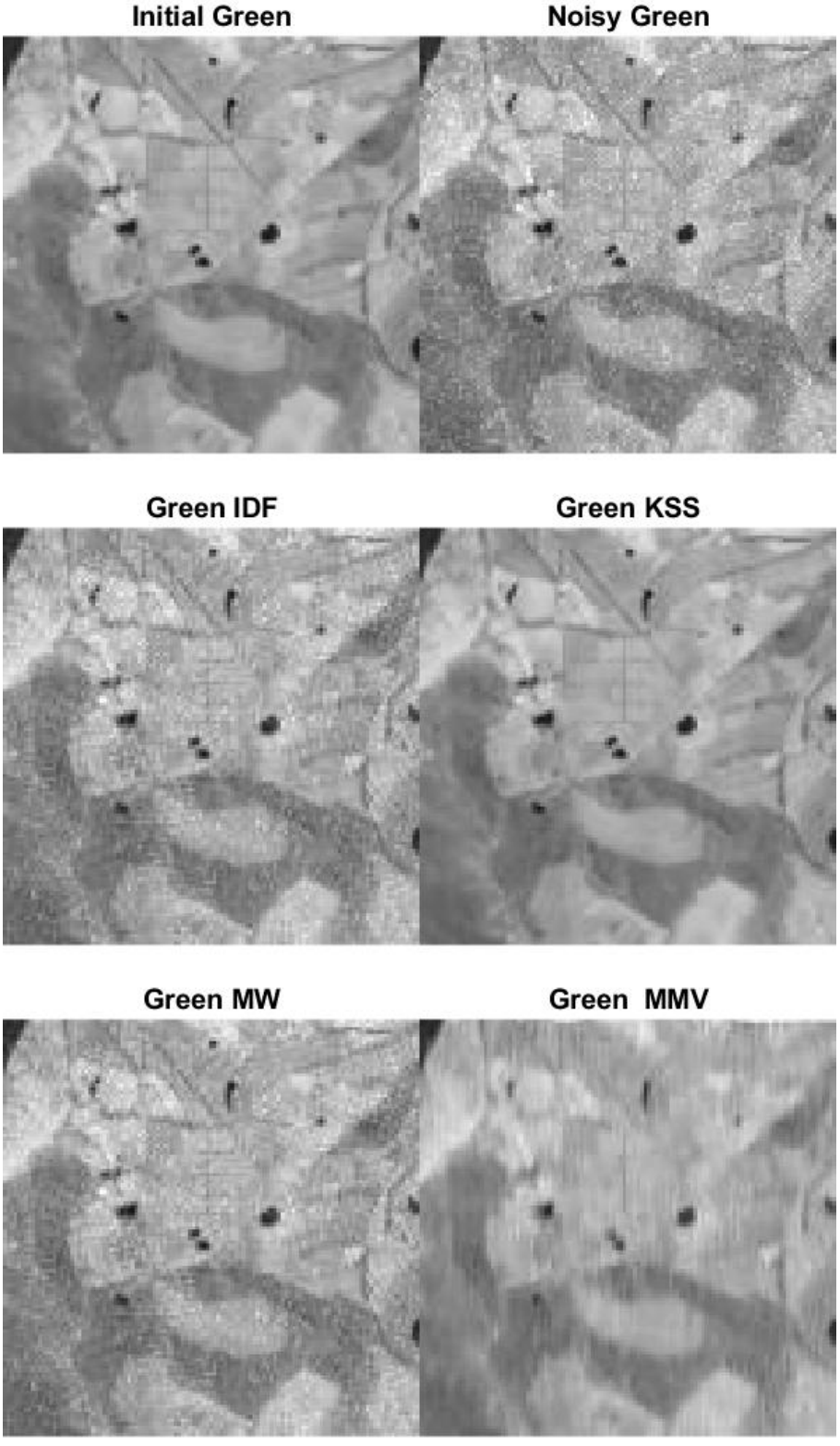

Figure 4.20: Landsat Green Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right),and MMV and MW (bottom left and right)

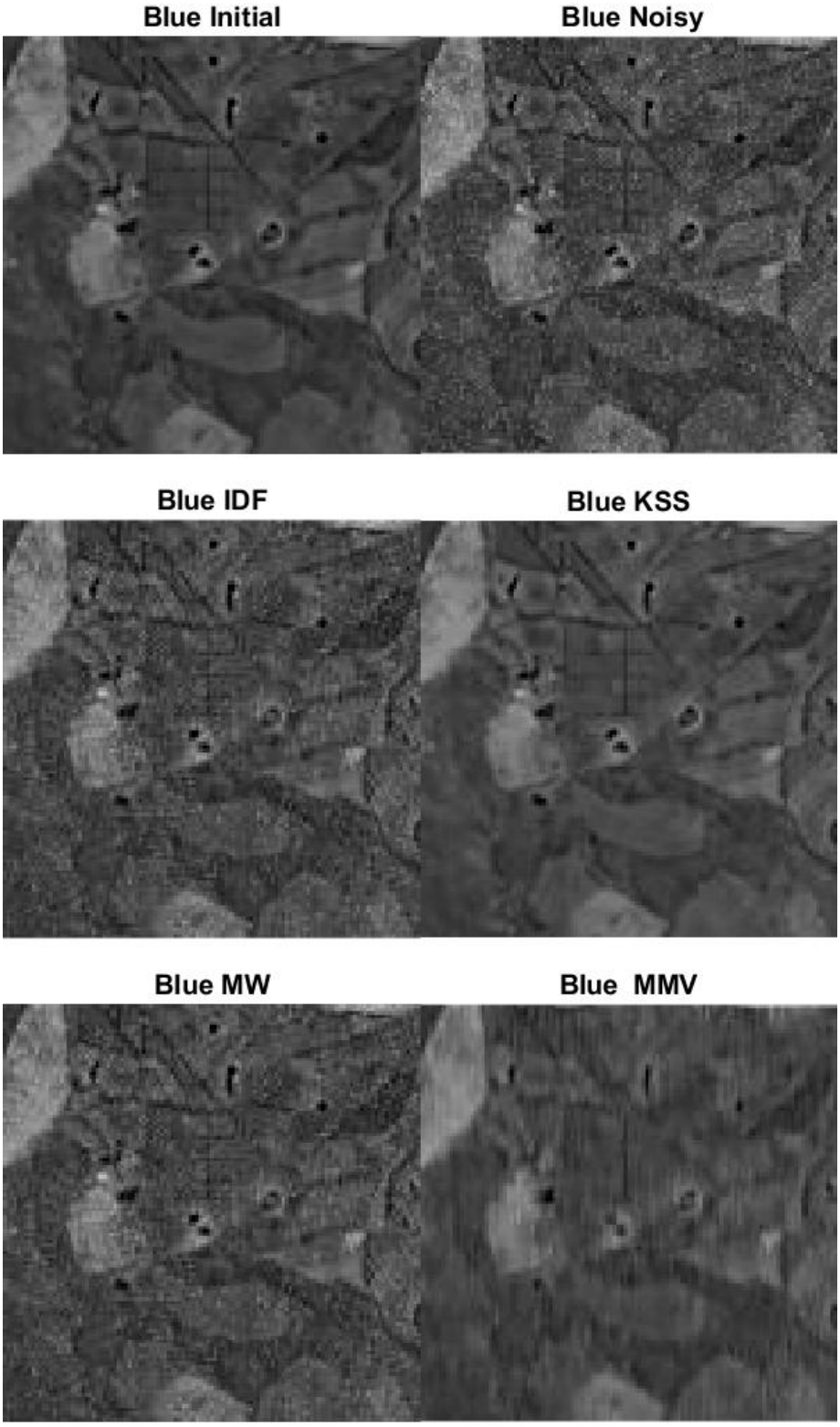

Figure 4.21: Landsat Blue Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figures in this section shows the Landsat image in initial, noisy $n = 0.0044$, and denoised states as whole images in Figures 4.13 to 4.17, block images in Figure 4.18 and cropped images in Figures 4.19 to 4.22. Once again in all images the denoising is successful for the KSS method in comparison to all other methods. KSS has removed the noise and doesn't have added blur as the regularization parameter is able to sharpen the image. The comparison methods IDF, MW have lingering noise and graininess but less than in the prior section as the noise is reduced and the MMV has some blur.

4.2.3 Peppers Test Results Noise 0.010

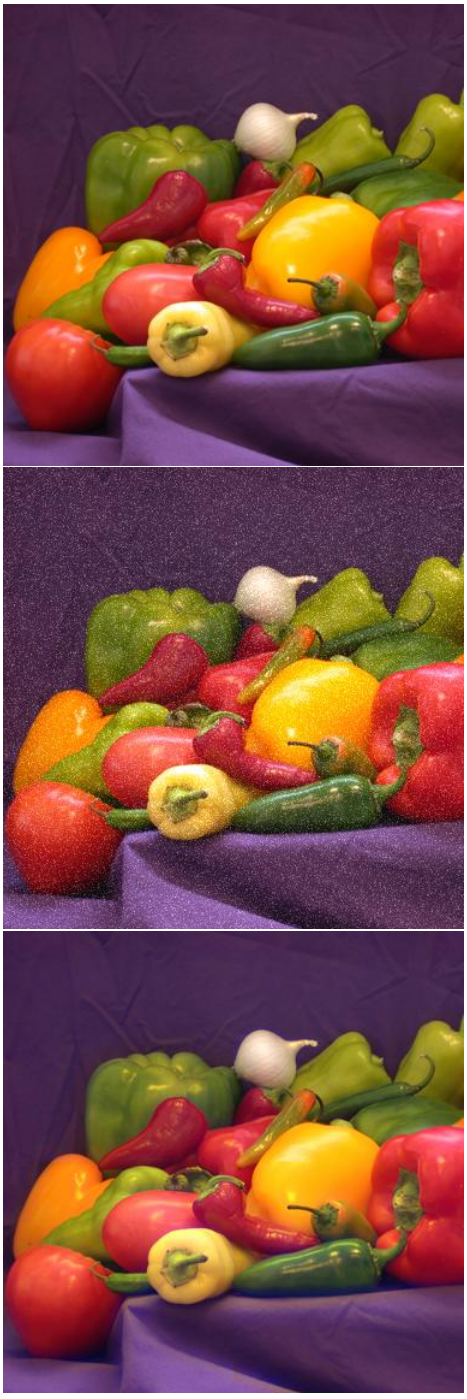

Figure 4.22: Peppers Whole Initial (top) Noisy $n = 0.010$ (middle) and KSS Denoised (bottom) RGB images

Figure 4.23: Peppers Whole Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.010 noise

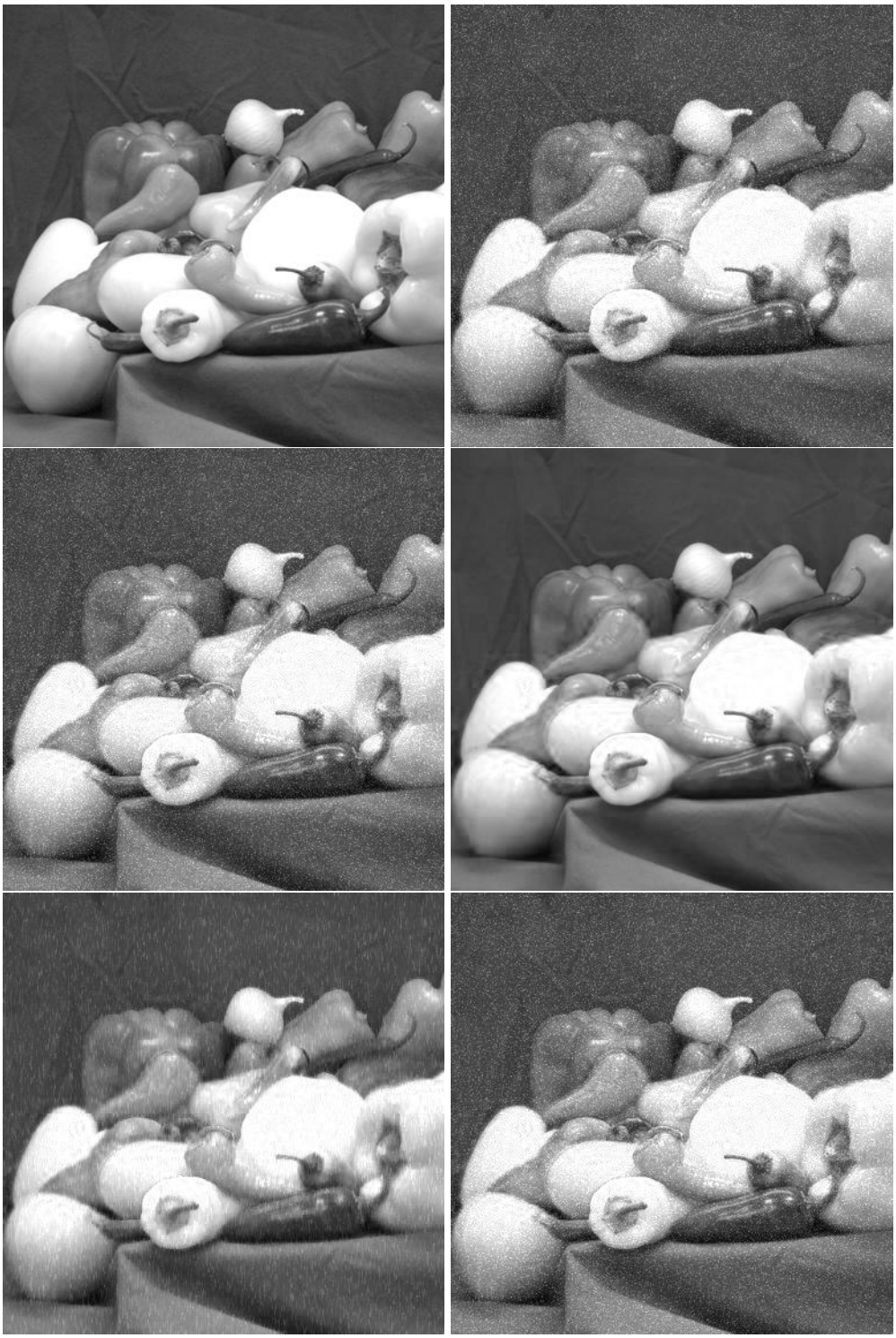

Figure 4.24: Peppers Whole Red Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

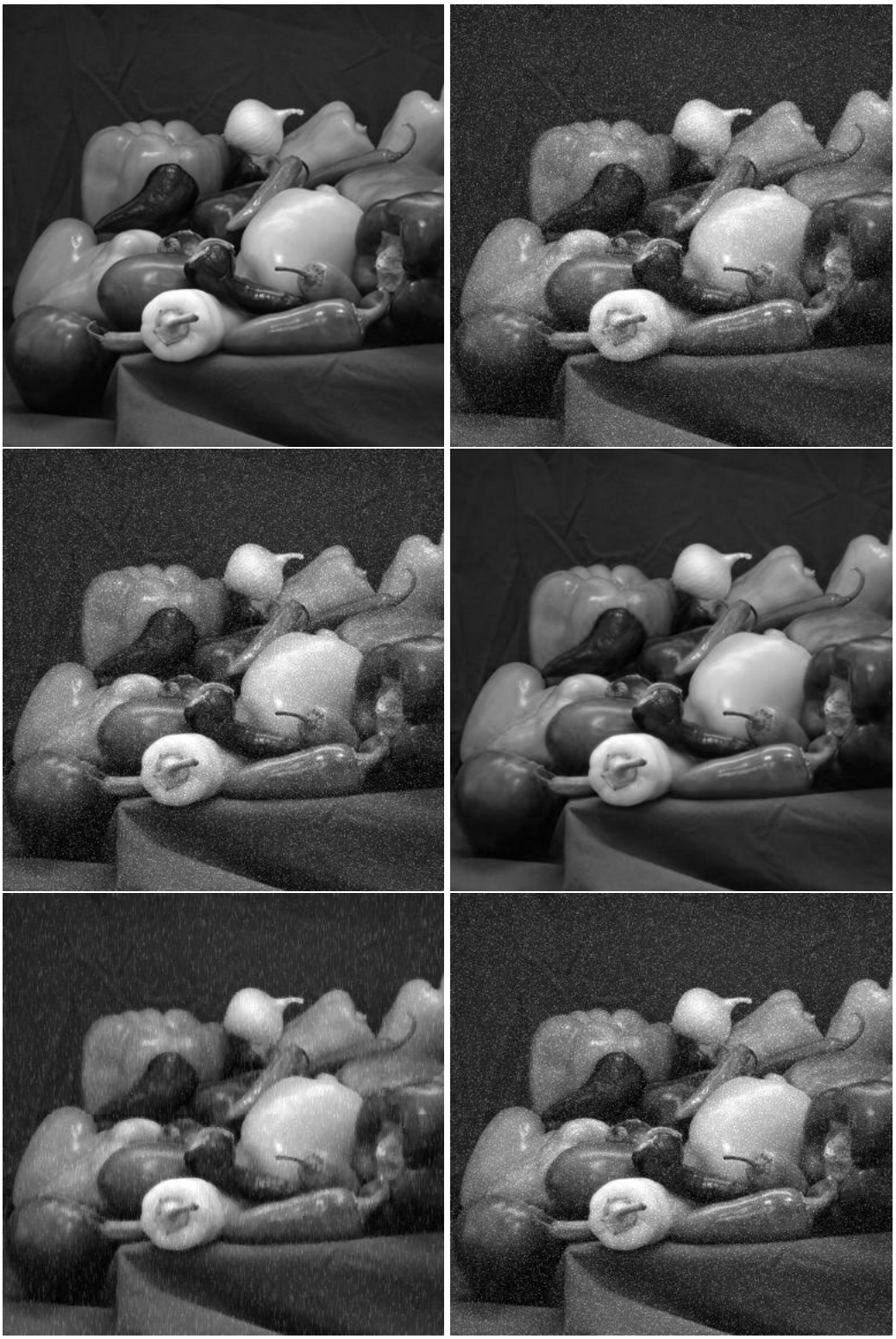

Figure 4.25: Peppers Whole Green Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

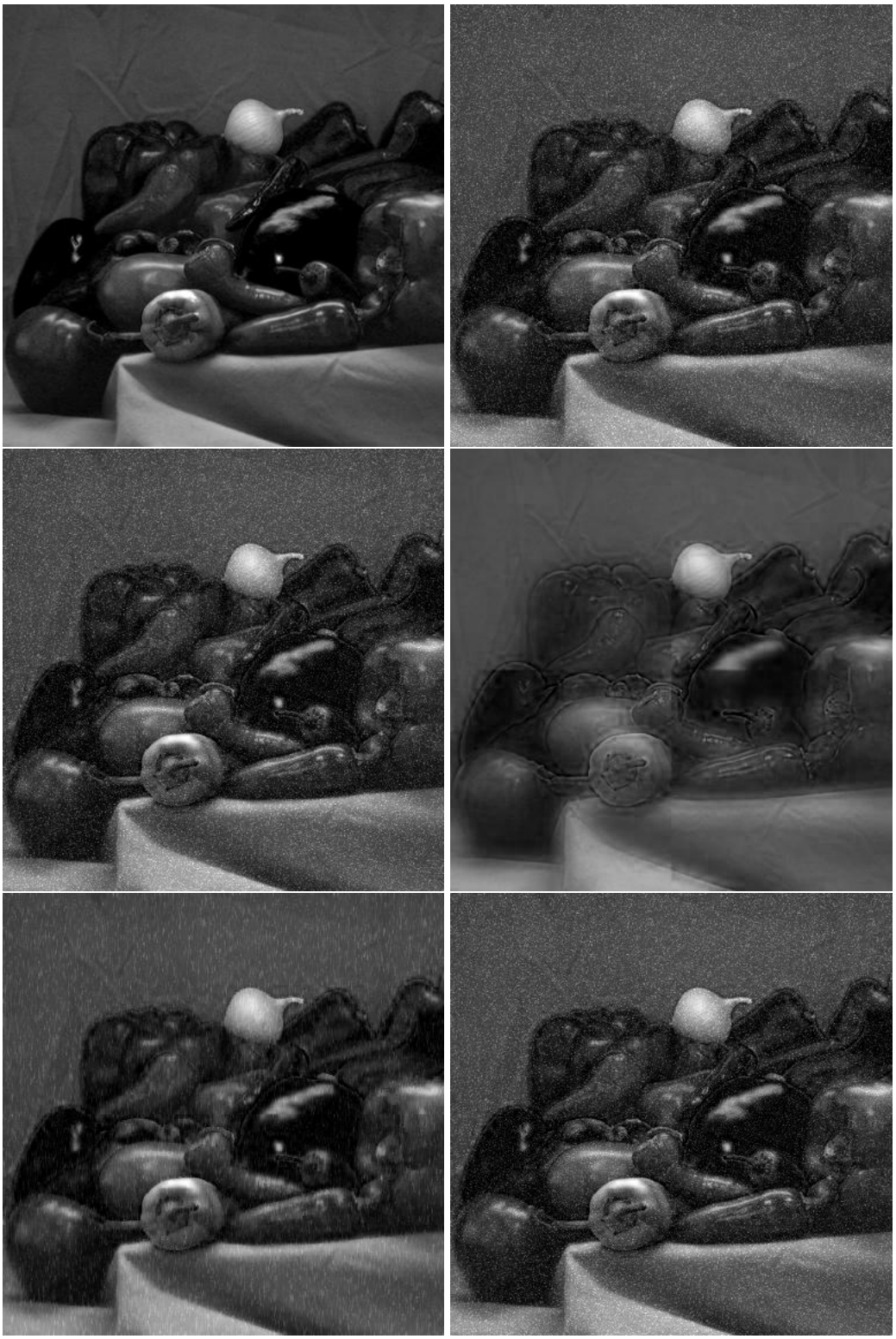

Figure 4.26: Peppers Whole Blue Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

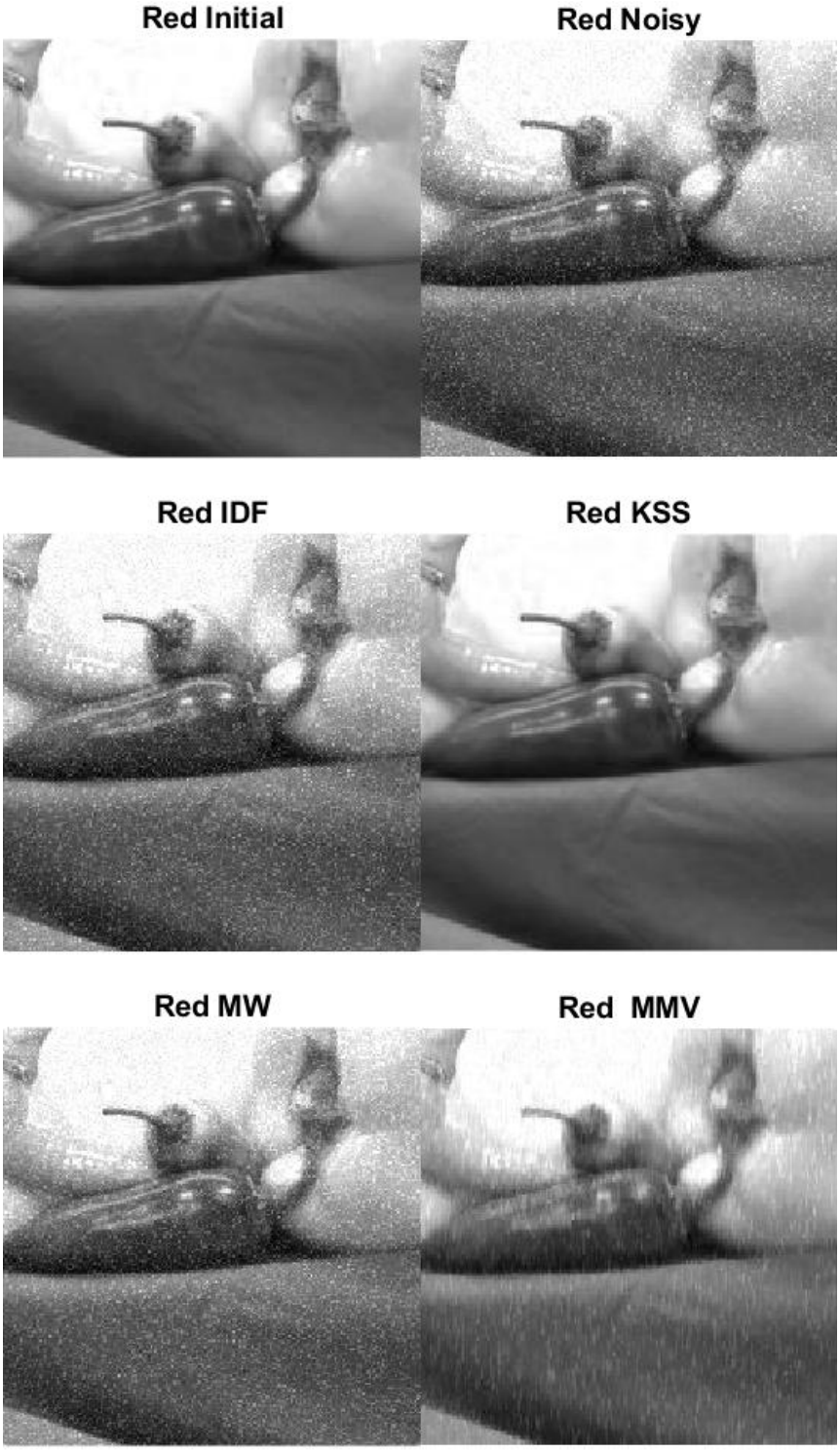

Figure 4.27: Peppers Red Block 22 Images: Initial and Noisy Images $n = 0.010$ (top left and right), and Images Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

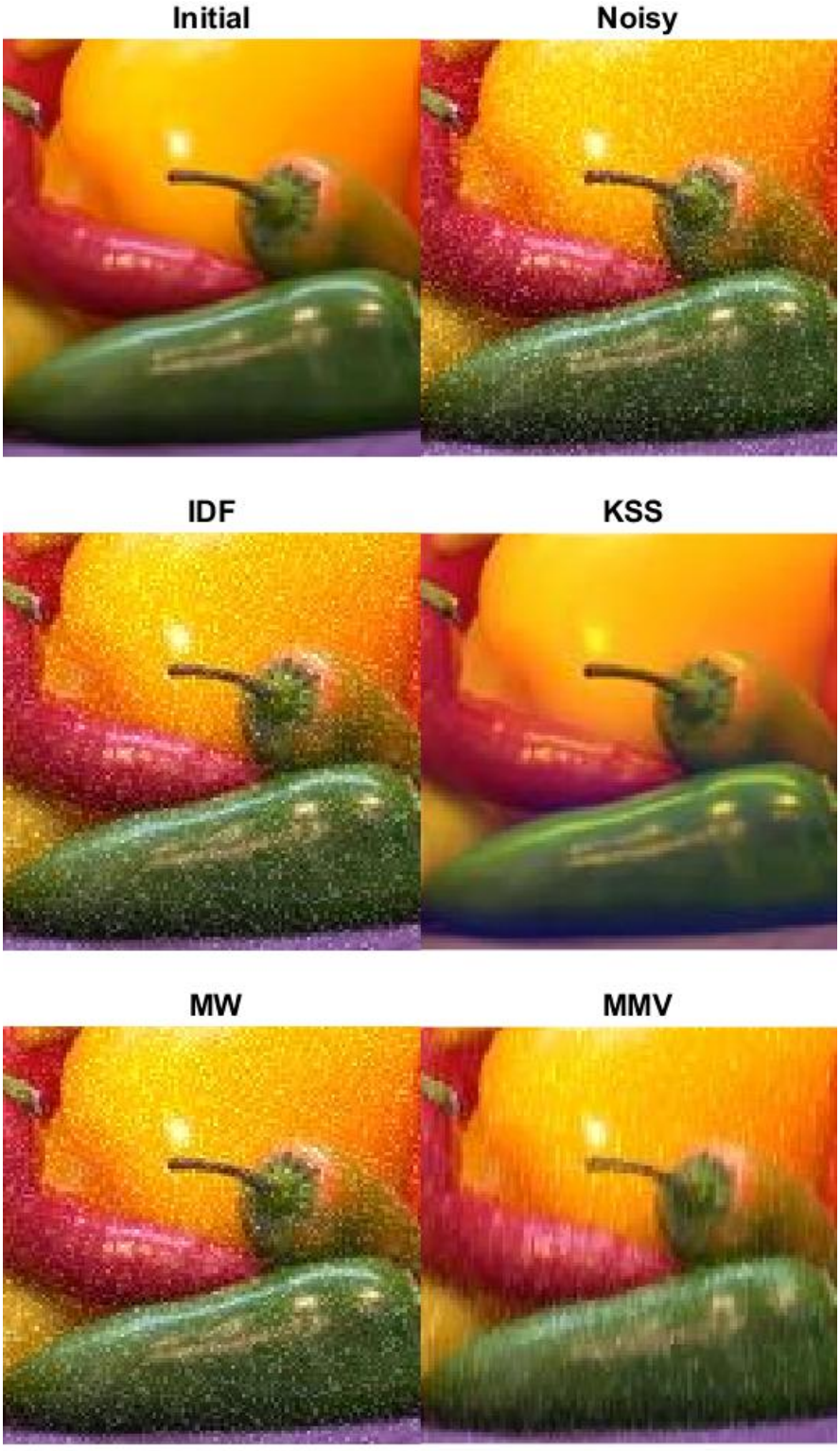

Figure 4.28: Peppers RGB Cropped Images: Initial and Noisy Images $n = 0.010$ (top left and right), and Images Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

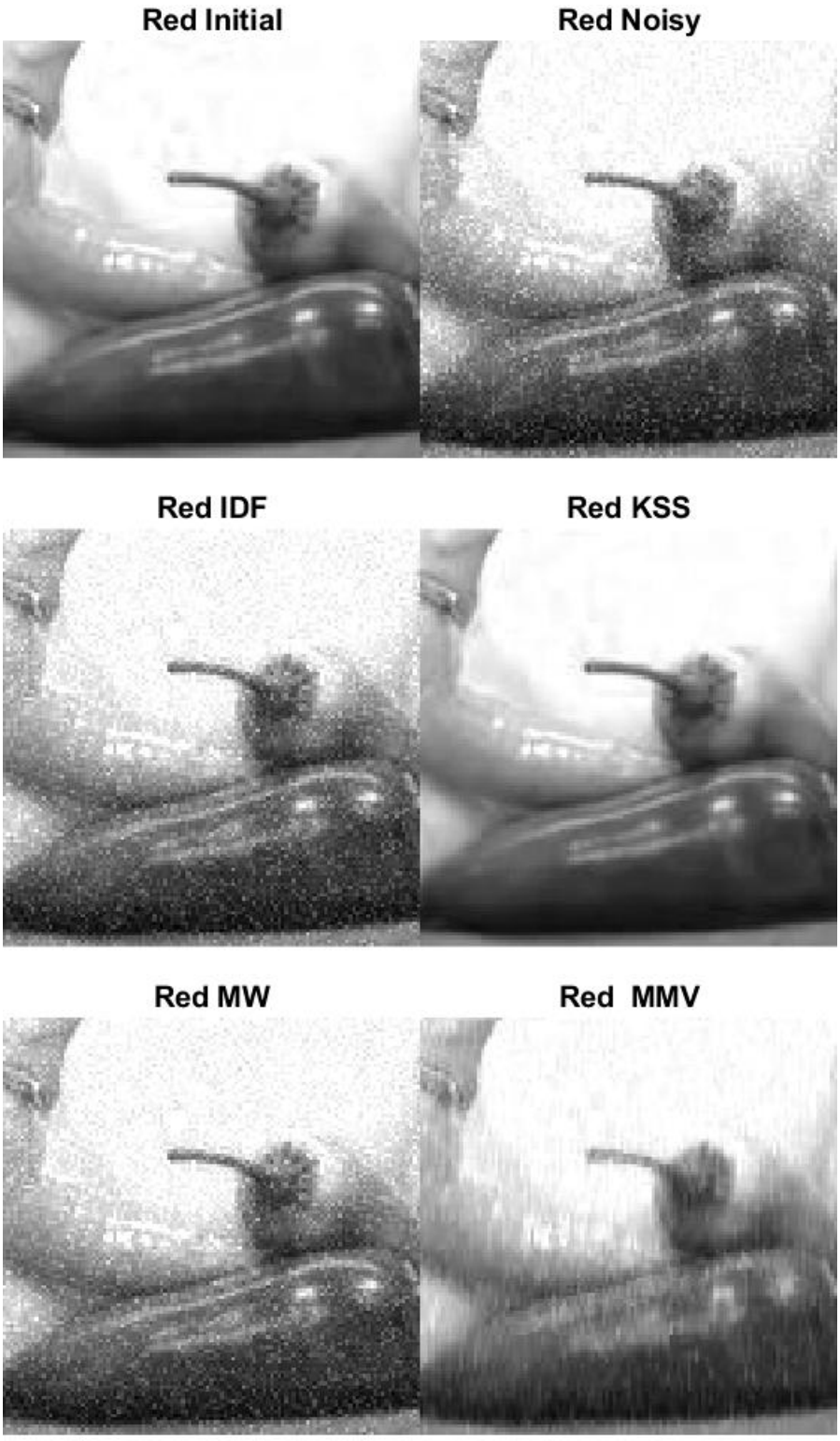

Figure 4.29: Peppers Red Cropped Images: Initial and Noisy Images $n = 0.010$ (top left and right), and Images Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

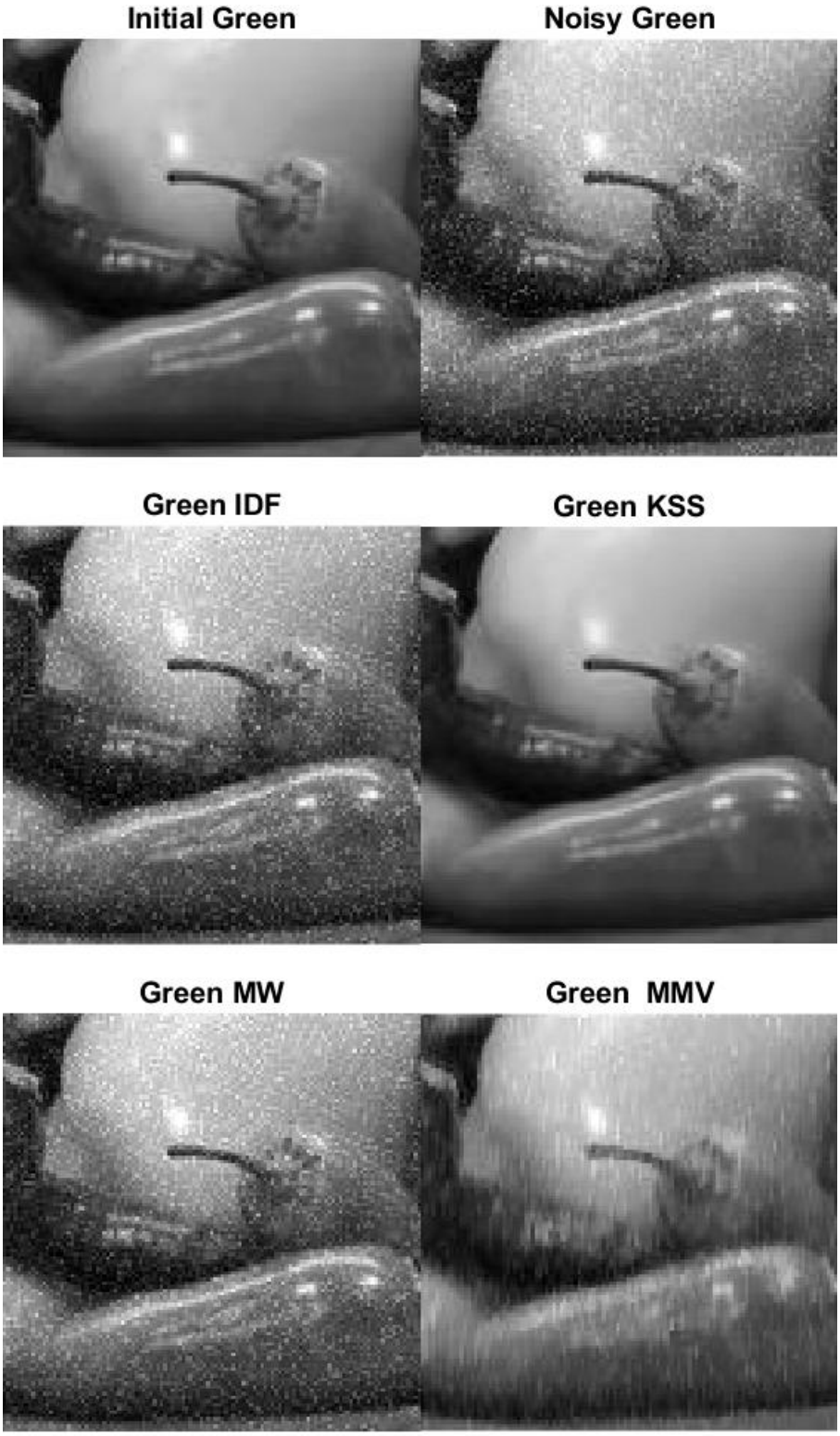

Figure 4.30: Peppers Green Cropped Images: Initial and Noisy Images $n = 0.010$ (top left and right), and Images Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

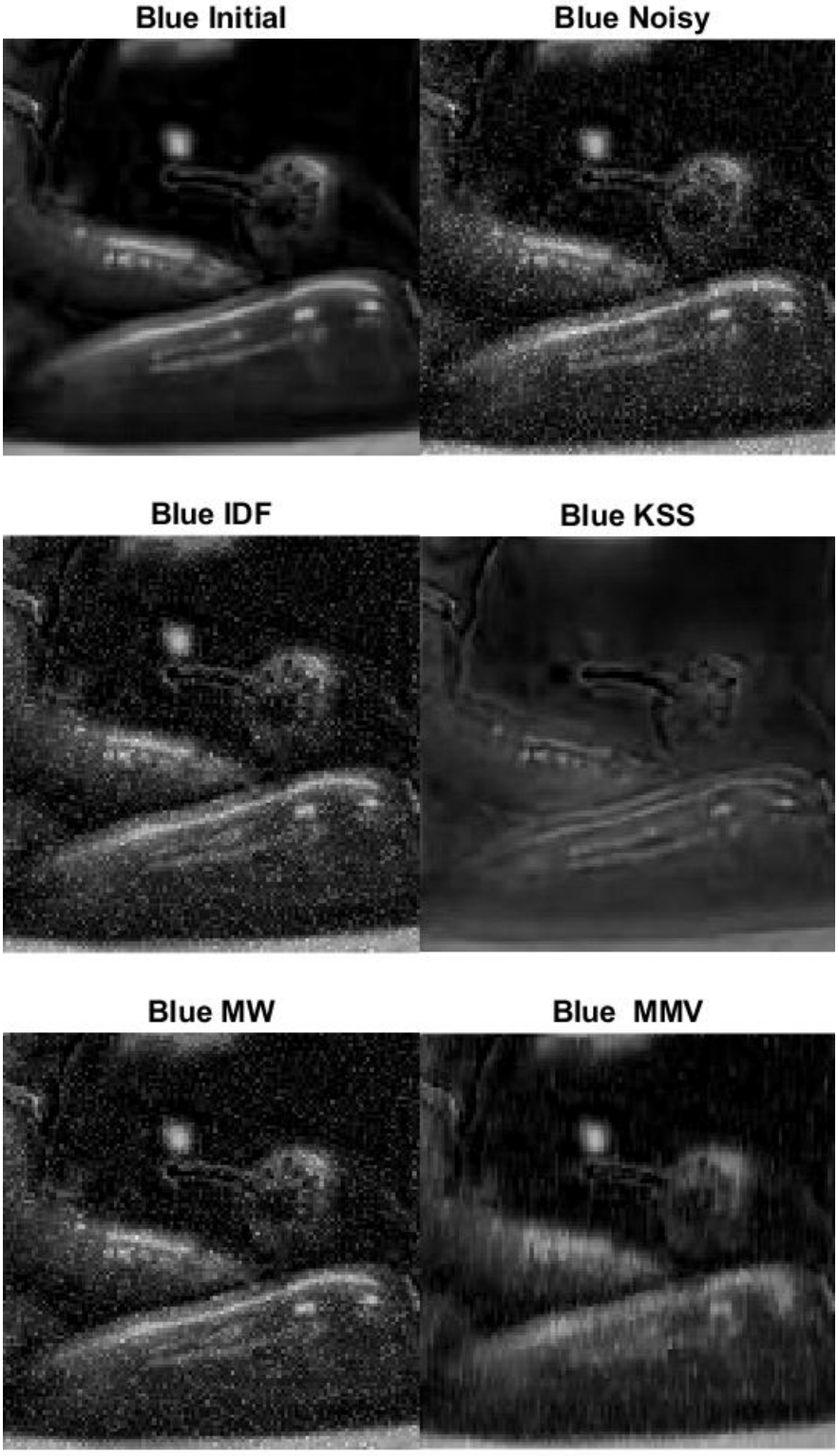

Figure 4.31: Peppers Blue Cropped Images: Initial and Noisy Images $n = 0.010$ (top left and right), and Images Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

In this section the Peppers image with $n = 0.010$ is presented. This image is different from the Landsat image in that its focal distance is for a subject much closer than the satellite image. It has less inherent noise in the image, as there is not atmospheric, dust or light diffraction in it as there is most probably in the satellite image. It is also contains very little blue color and has a dark background. Figures in this section show the Peppers image in initial, noisy $n = 0.010$, and denoised states as whole images in Figures 4.23 to 4.27, block images in Figure 4.28 and cropped images in Figures 4.29 to 4.32. Figure 4.24 shows the whole denoised blue image in the lower right, which is very poor quality and has visible lines between the four subblocks, which result from the varying final states of denoising. In Figures 4.25 and 4.26 all denoising methods are compared in the red and green images. The KSS and IDF methods denoise very well, but the MMV, MW methods are grainy and have remaining noise. In Figure 4.27, the KSS denoising has very poor results with extreme blurring in all but the upper right subblock. The IDF, MMV and MW methods have remaining noise and graininess in the blue image as well. Recall the peppers image contains little blue , so its pixel intensity for the blue images is much less than in the red and green images. This researcher hypothesizes that the presence of the white onion in the upper right block of the image has raised the pixel intensity in the blue image so that denoising of that block is more successful than in the other blocks for the blue image.

In Figure 4.32 the cropped images are difficult to interpret. Looking at the initial image shows a hardly recognizable peppers image becuase only reflection and highlighted edges can be seen. The noisy image is more recognizable because the noise makes the surfaces of the peppers more visible. The KSS image shows effective denoising, but it has left a ghost like appearance from added blur. The IDF, MW and MMV denoising methods have noise remaining but no blur. It is ironic that the remaining noisy helps to see the image surfaces. One could incorrectly assume the IDF, MW and MMV images have a better overall quality, because the noise on the surface makes images more recognizeable. The best denoising is determined not by your ease of seeing the peppers image, but by the lack of noise and or blur. The blue peppers image is a challenging one for all denoising methods. It is interesting that the presence of some noise does actually help human perception of three dimensional surfaces in a single color. Examination of Figure 4.29 demonstrates that the blue blur is less important than the remaining noise, and KSS denoising is the most successful method.

4.2.4 Peppers Test Results Noise 0.0044

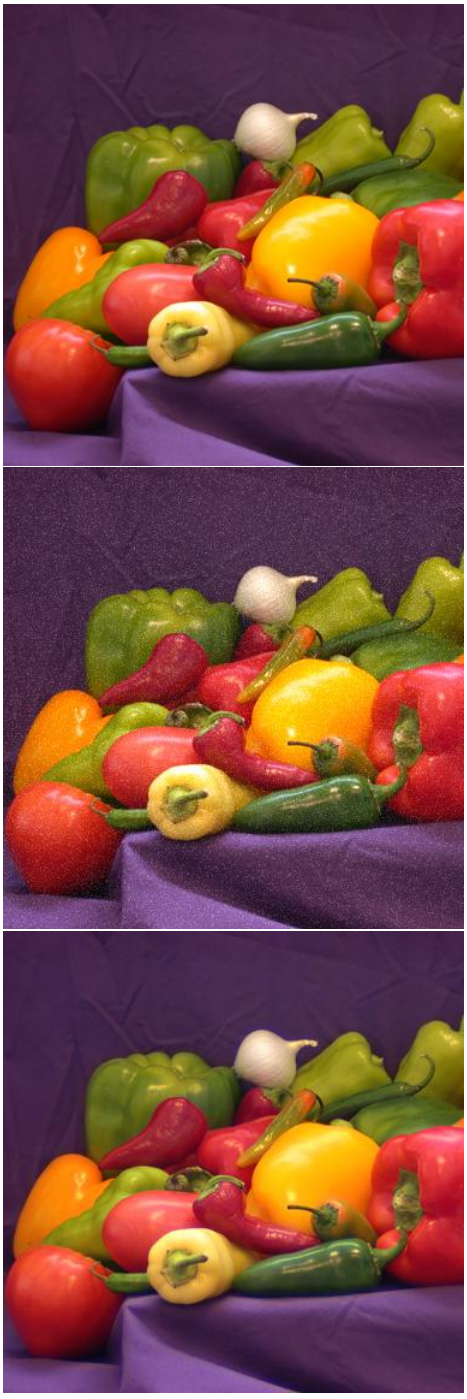

Figure 4.32: Peppers Whole Initial (top) Noisy $n = 0.0044$ (middle) and KSS Denoised (bottom) RGB images

Figure 4.33: Peppers Whole KSS Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.0044 noise

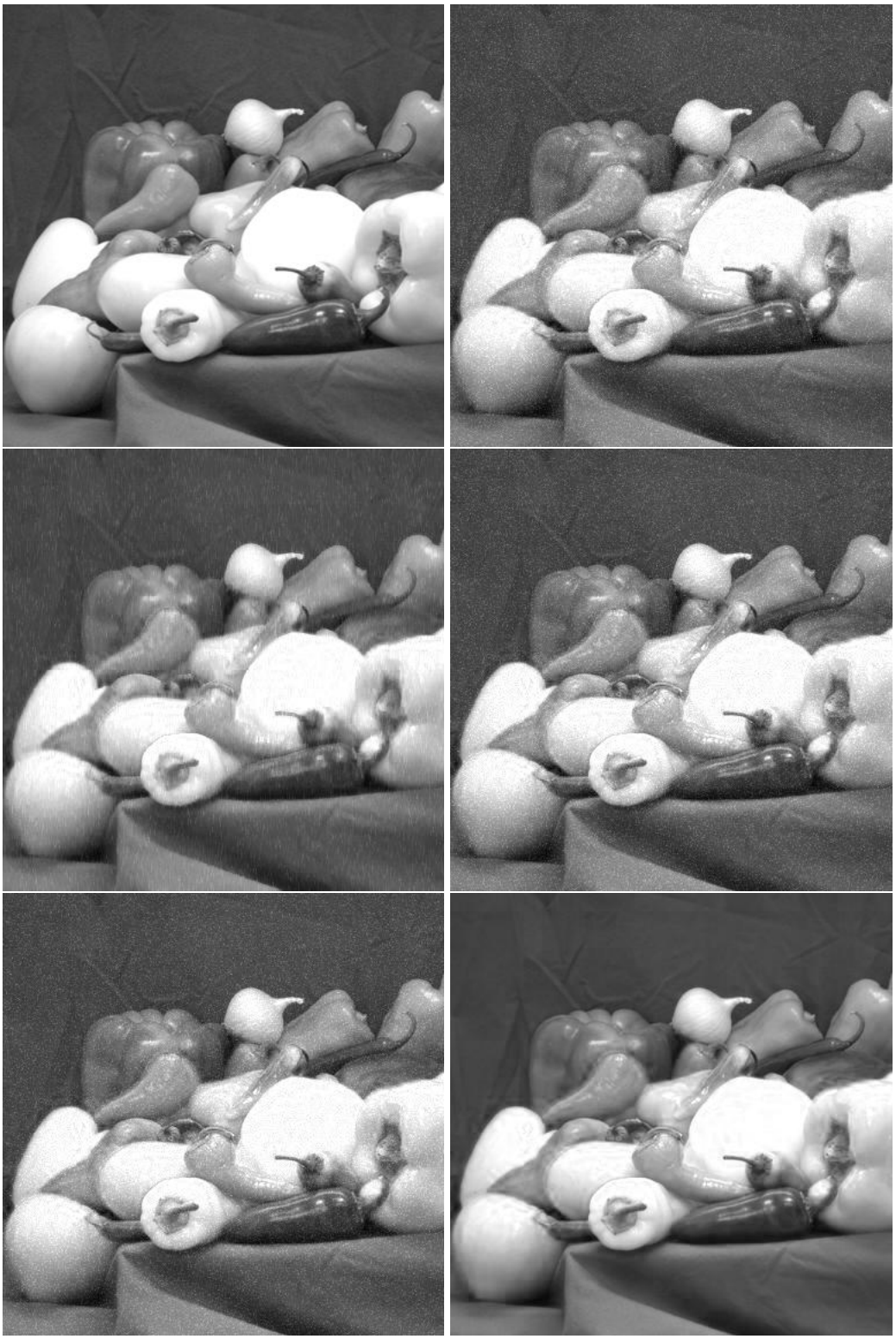

Figure 4.34: Peppers Whole Red Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

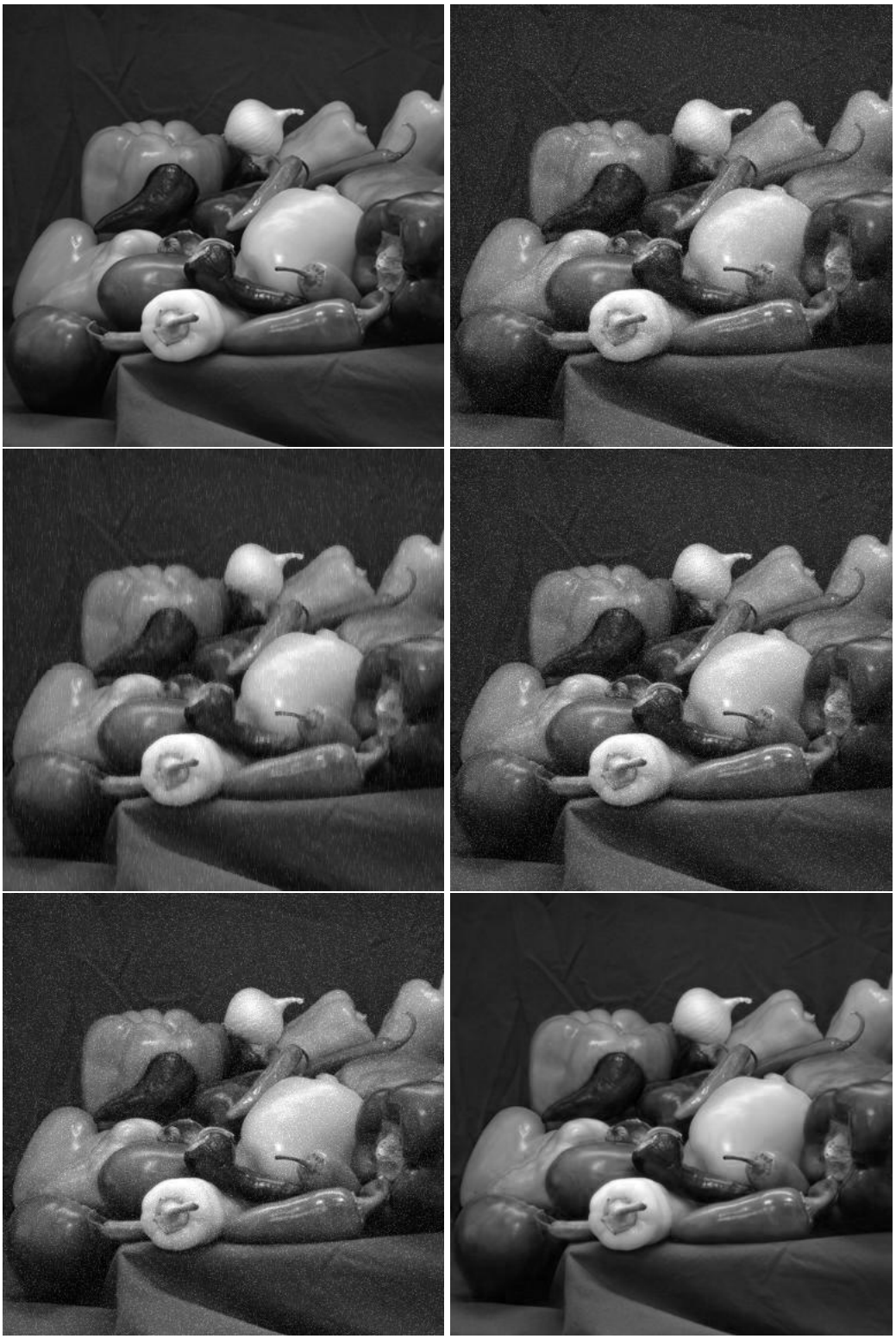

Figure 4.35: Peppers Whole Green Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

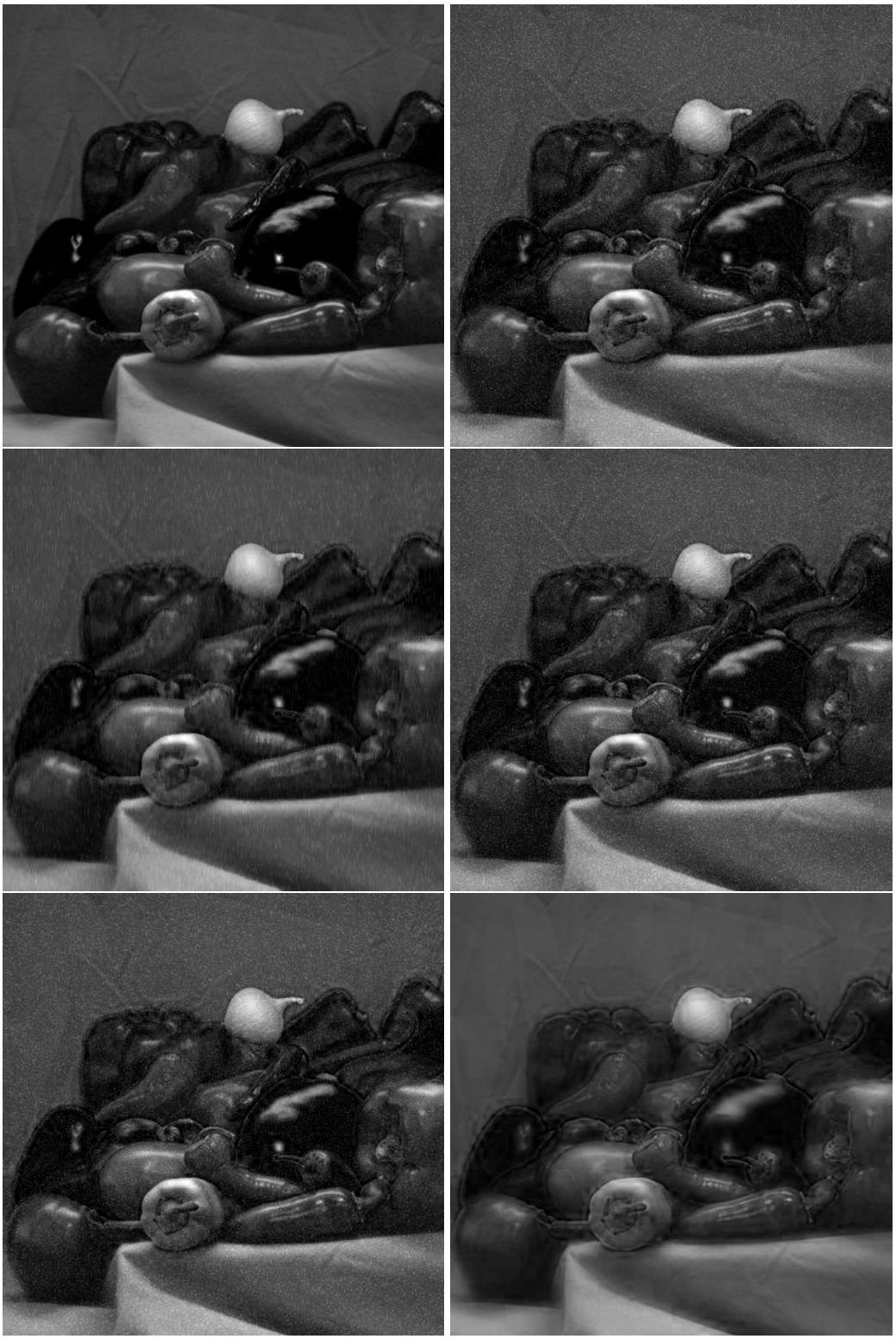

Figure 4.36: Peppers Whole Blue Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

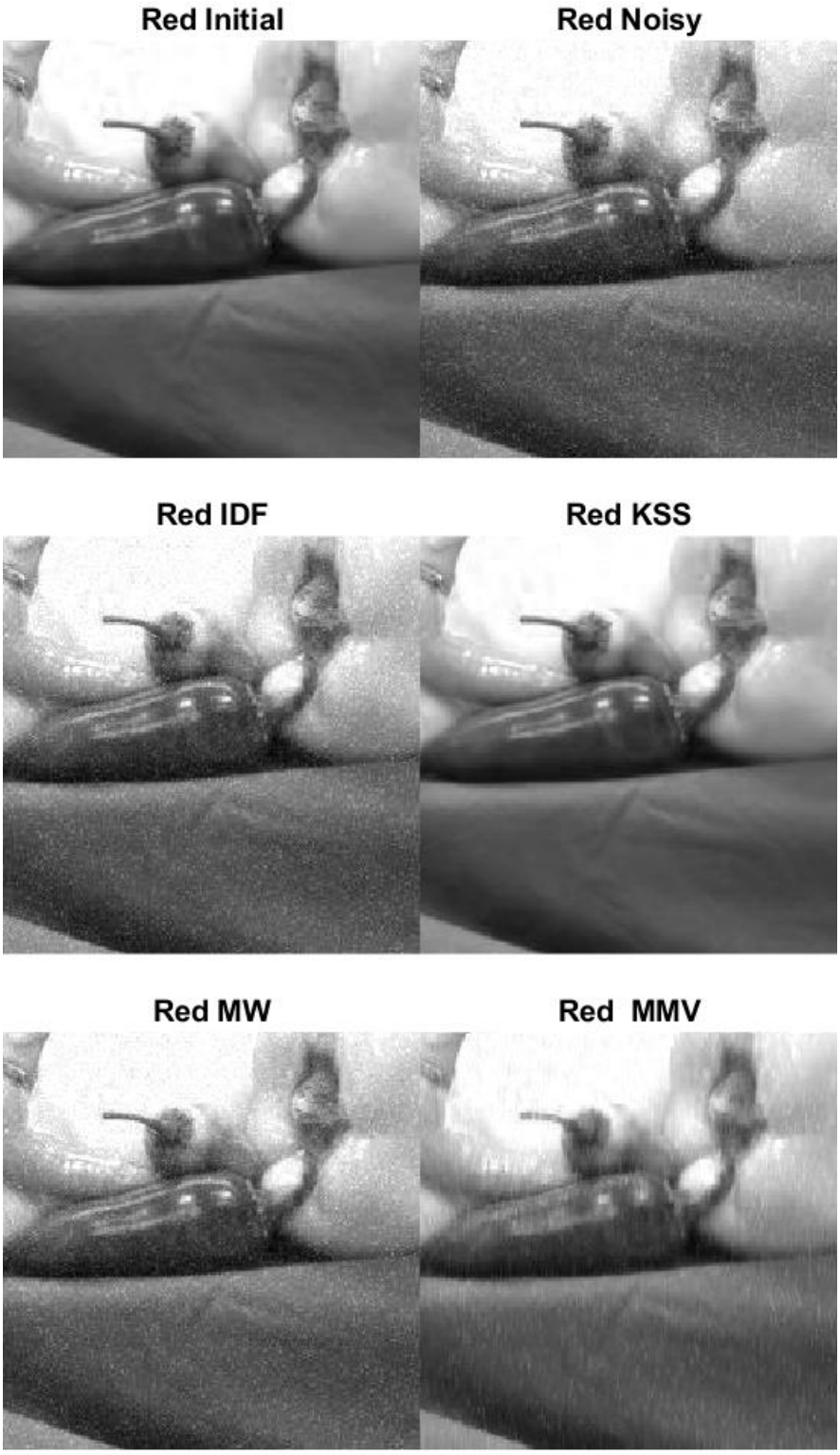

Figure 4.37: Peppers Red Block 22 Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

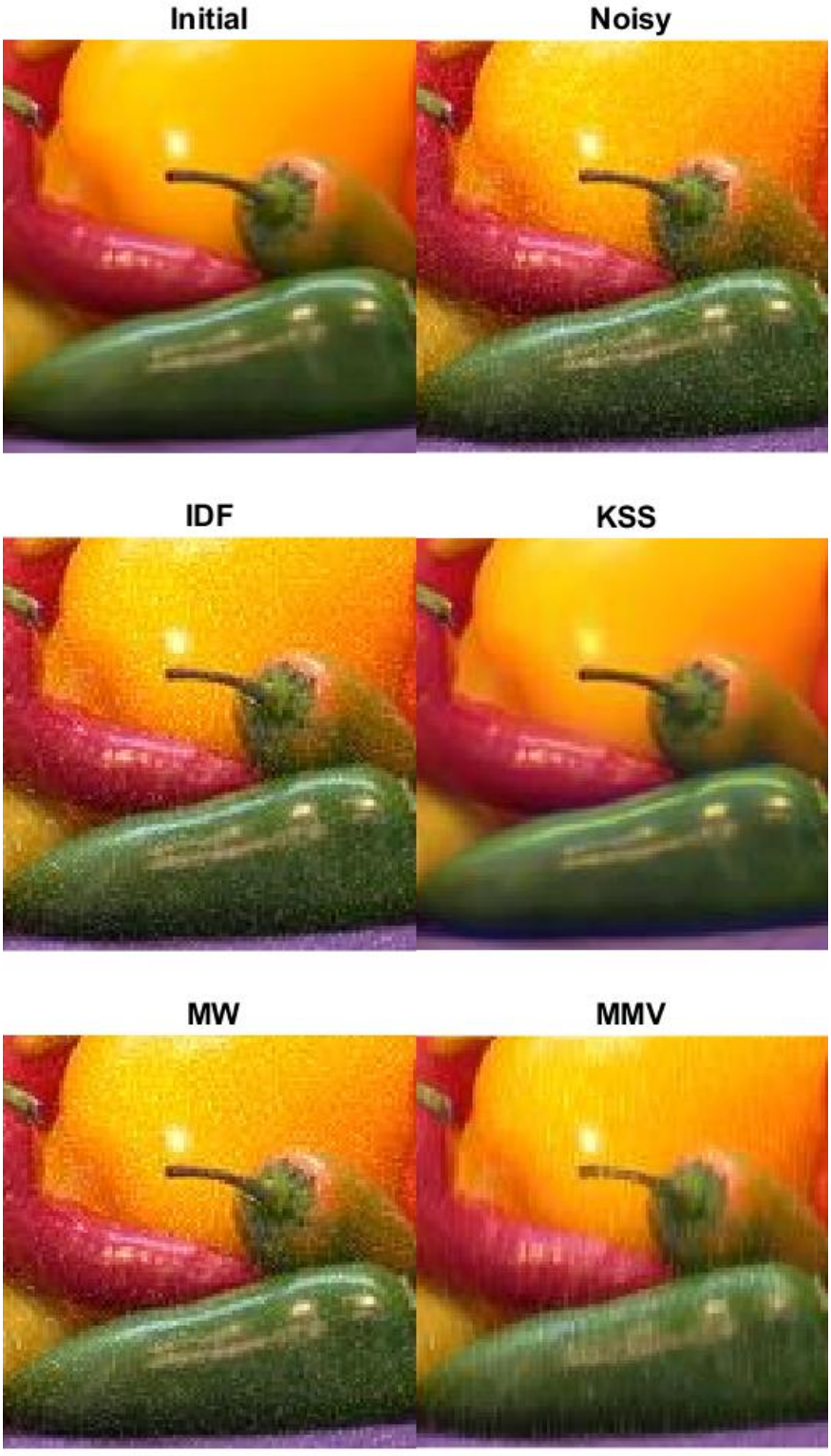

Figure 4.38: Peppers RGB Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

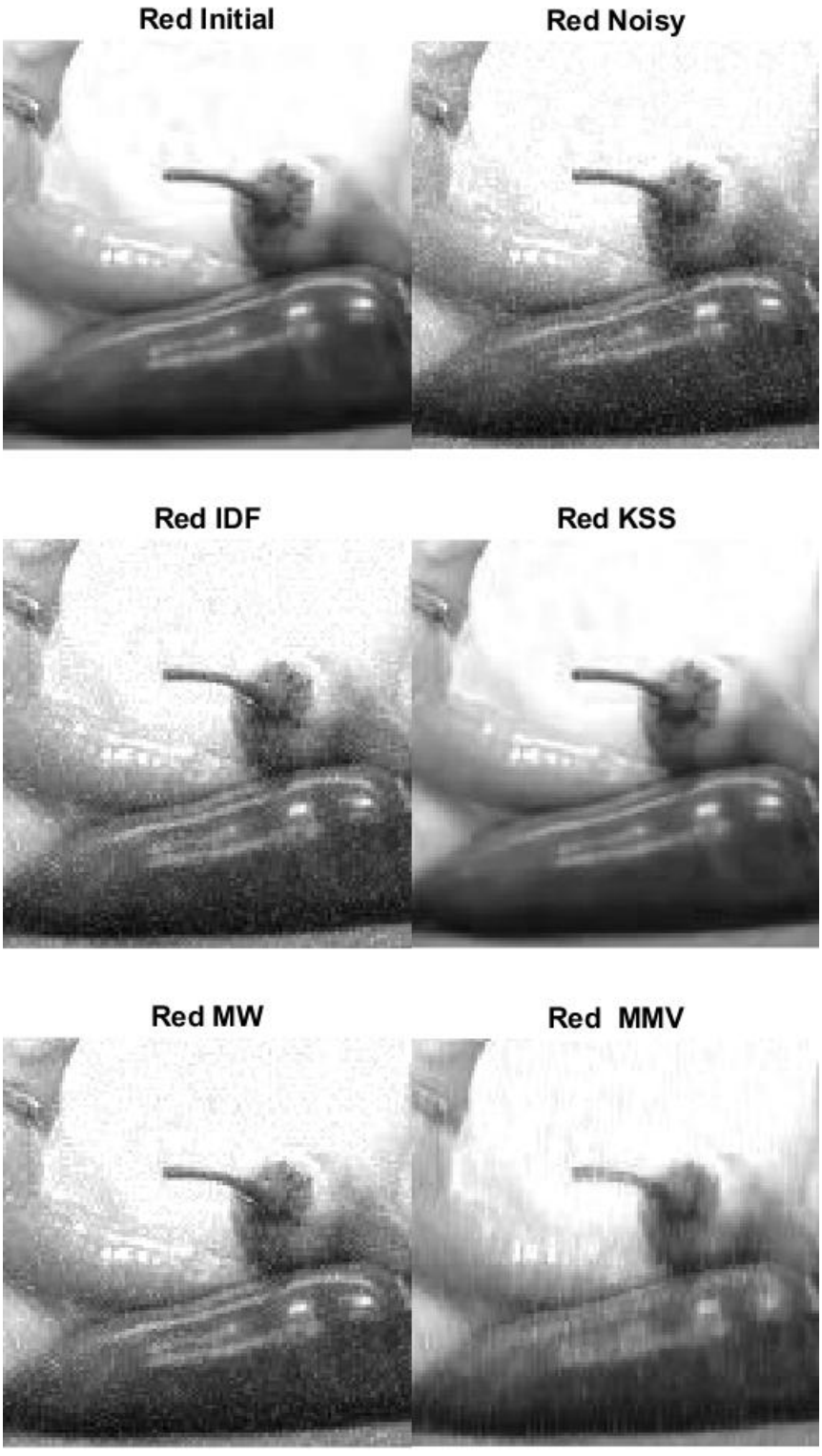

Figure 4.39: Peppers Red Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

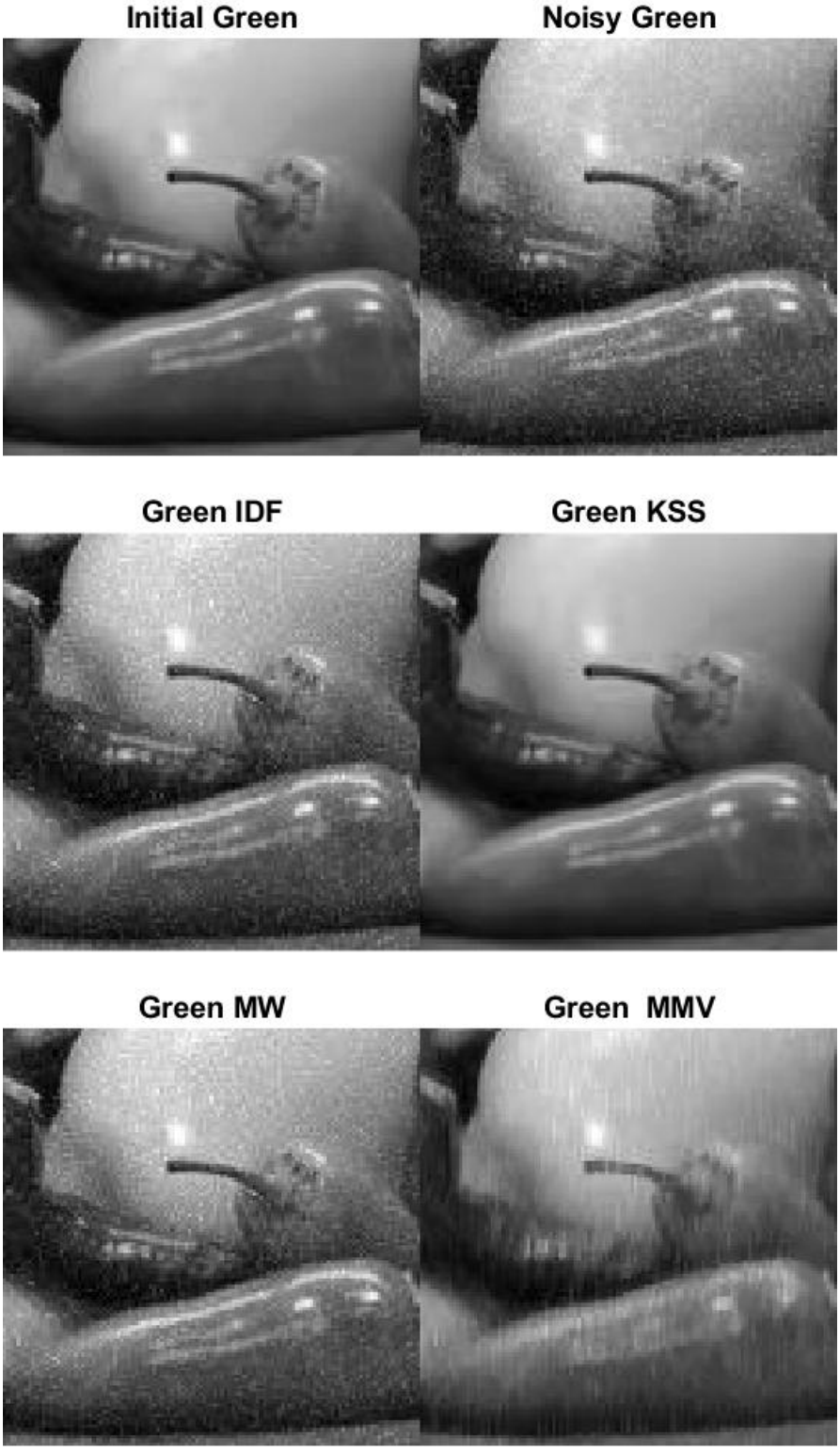

Figure 4.40: Peppers Green Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

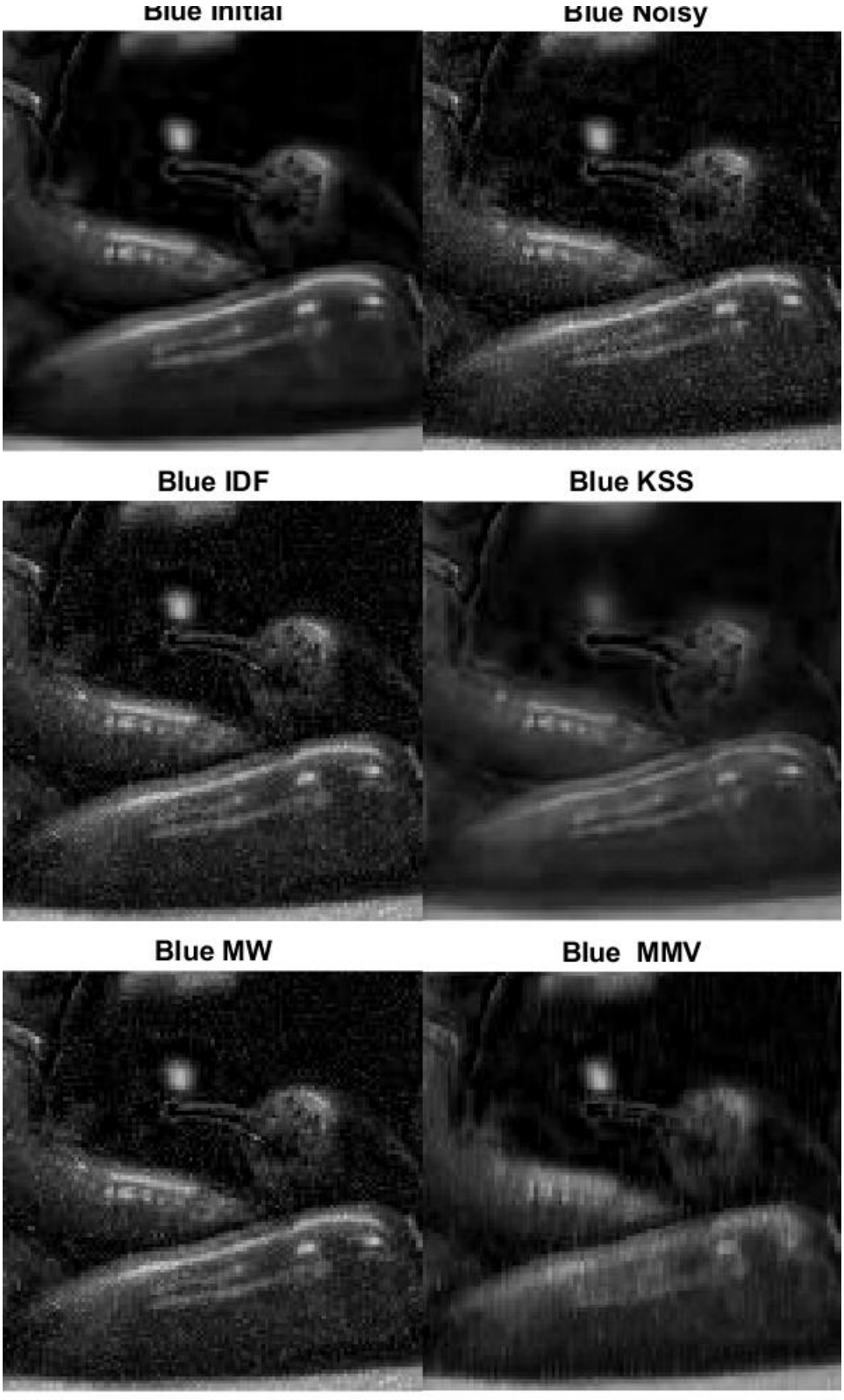

Figure 4.41: Peppers Blue Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

Figures in this section show the Peppers image in initial, noisy $n = 0.0044$, and denoised states as whole images in Figures 4.33 to 4.37, block images in Figure 4.38 and cropped images in Figures 4.39 to 4.42. Figure 4.34 shows the whole denoised blue image in the lower right. In Figure 4.34 the lower right sub figure there are only faintly visible lines between the blocks with the lower noise level as compared to the same in Figure 4.24 having the higher noise level. This is because with less noise, KSS denoising doesnt require as many timesteps and has less blur as a result. In Figure 4.42, the denoising of the cropped blue image is less blurry with the lower noise level and has no noise or graininess that is still present in the comparative methods. Overall the denoising is successful for the KSS method in comparison to all other methods. The comparison methods IDF, MW have lingering noise and graininess but less than in the prior section as the noise is reduced and the MMV has some blur. In Figure 4.39 the recombined denosied RGB cropped images show that KSS is the superior denoising method and the issues with blue blur are not noticeable.

4.2.5 Boat Test Results Noise 0.010

Figure 4.42: Boat Whole Initial (top) Noisy $n = 0.010$ (middle) and KSS Denoised (bottom) RGB images

Figure 4.43: Boat Whole Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.010 noise

Comparative Denoising Methods BiloxiS Noise 0.01

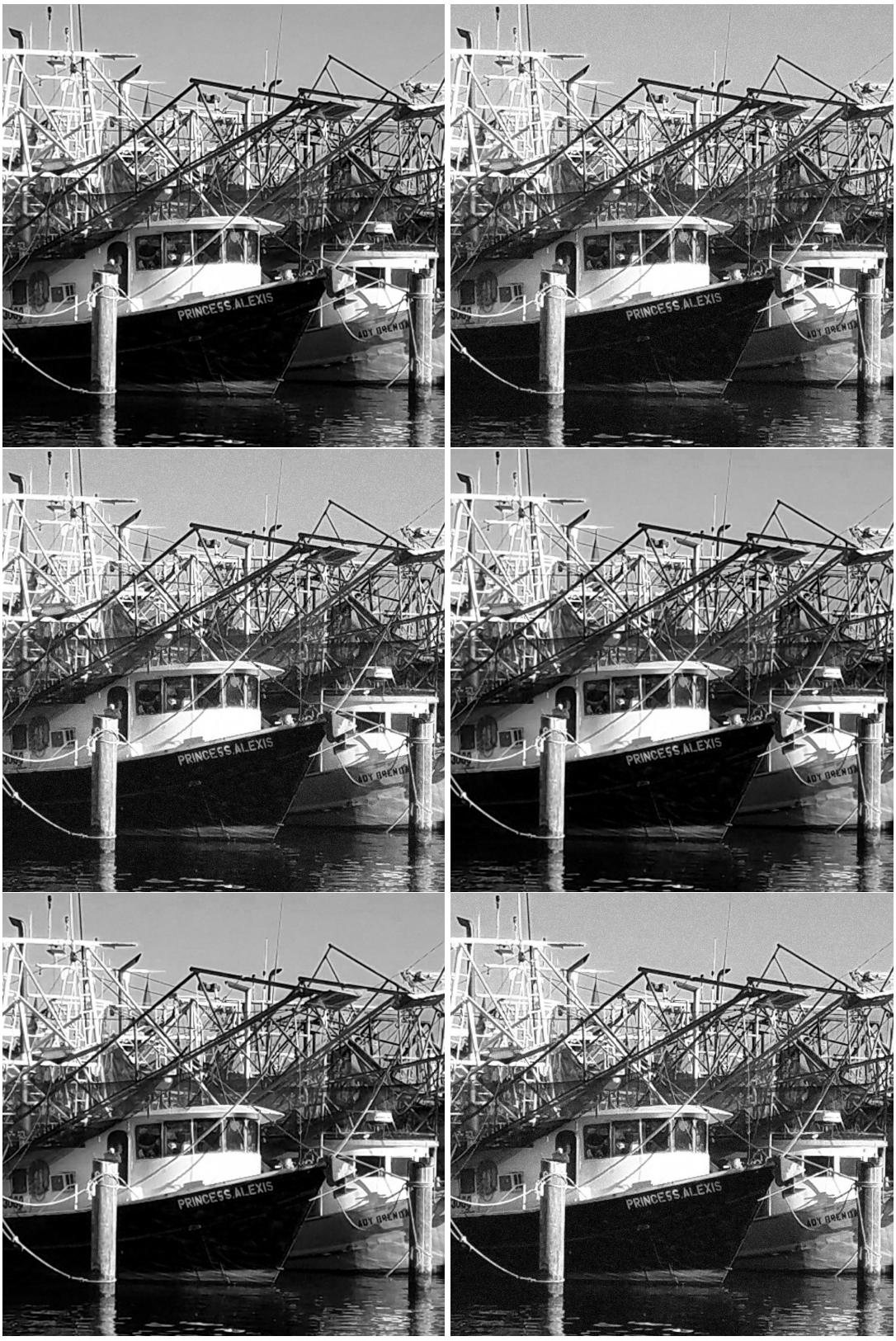

Figure 4.44: Boat Whole Red Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.45: Boat Whole Green Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.46: Boat Whole Blue Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

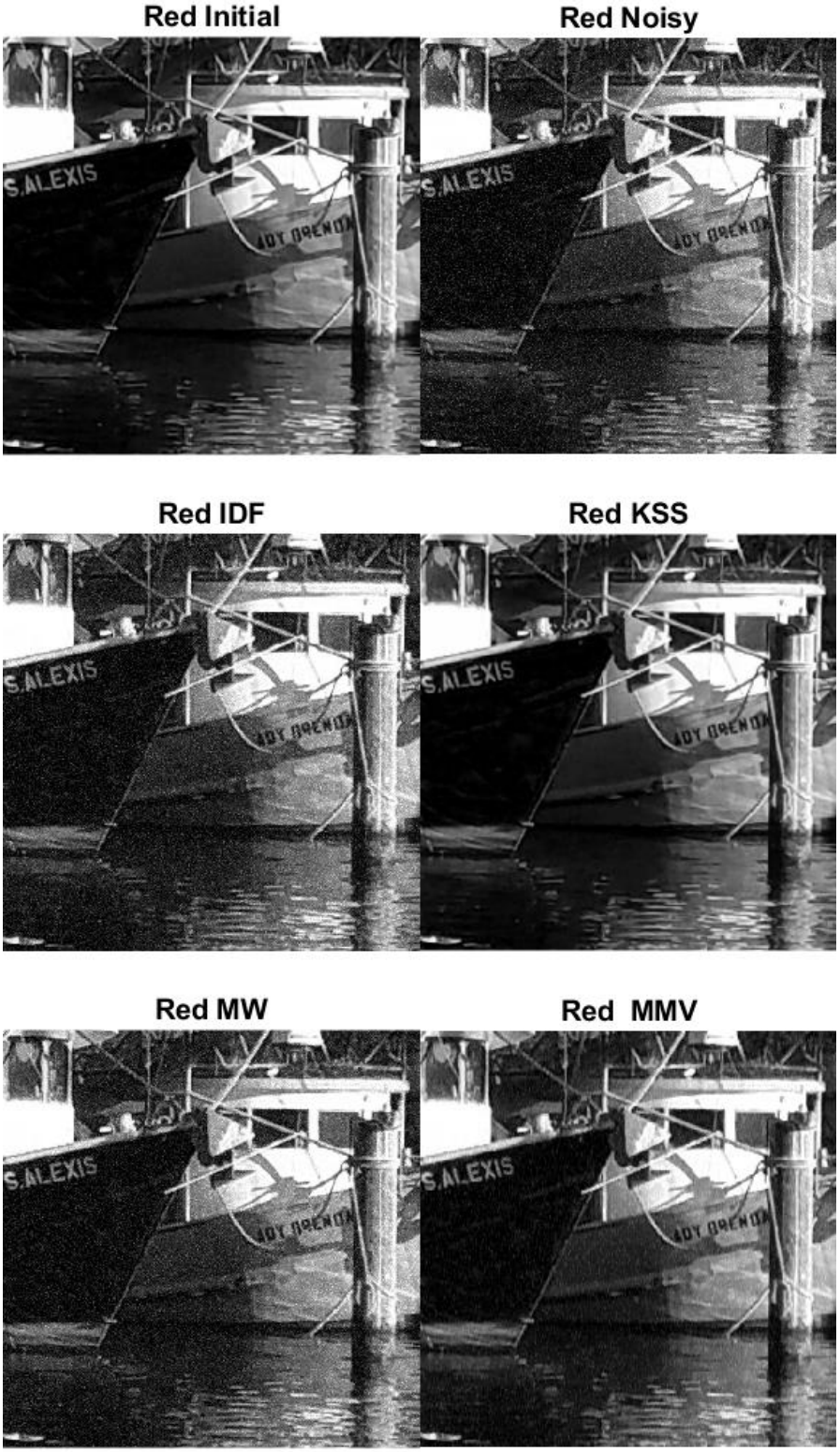

Figure 4.47: Boats Red Block 22 Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

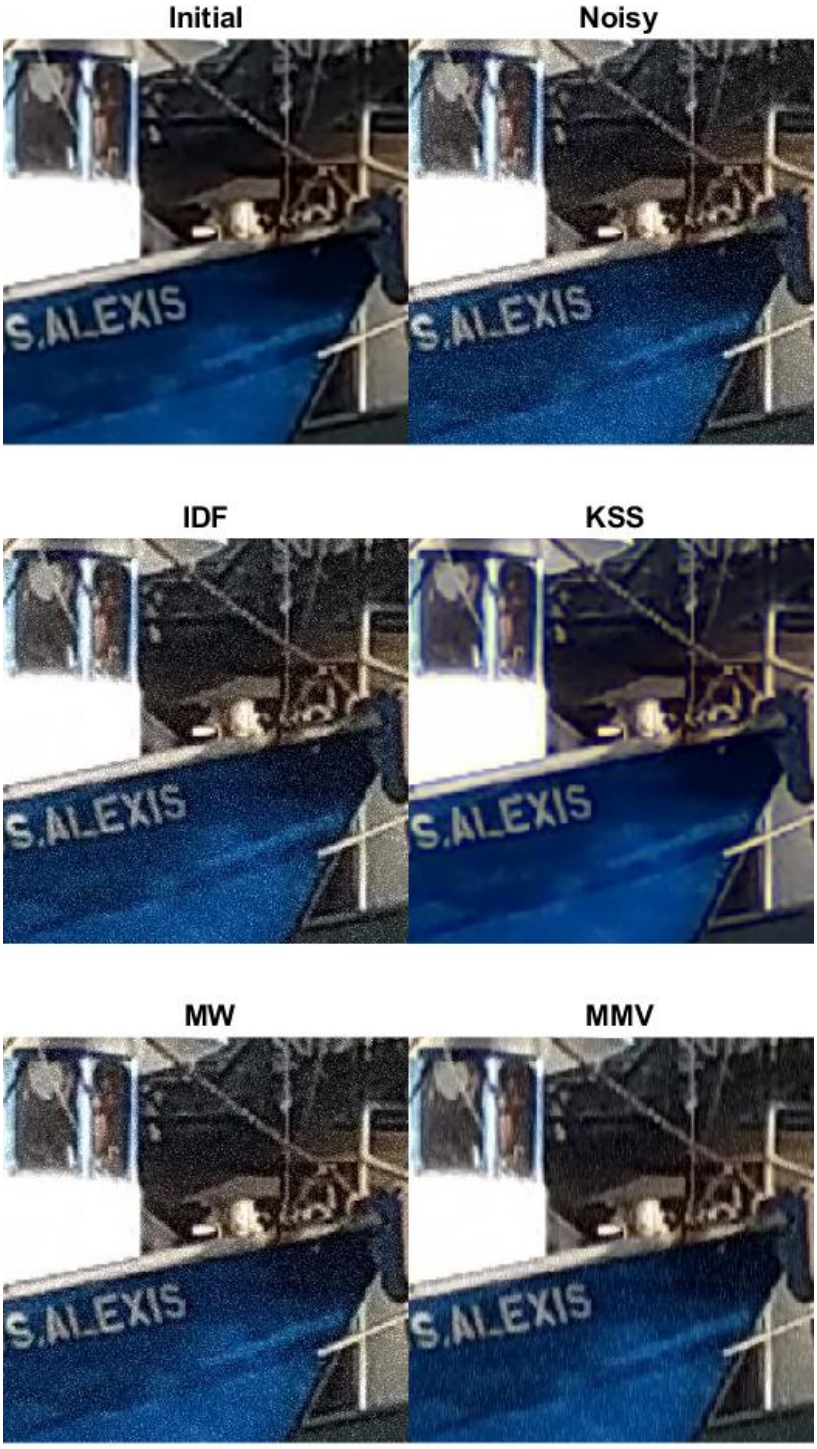

Figure 4.48: Boat RGB Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

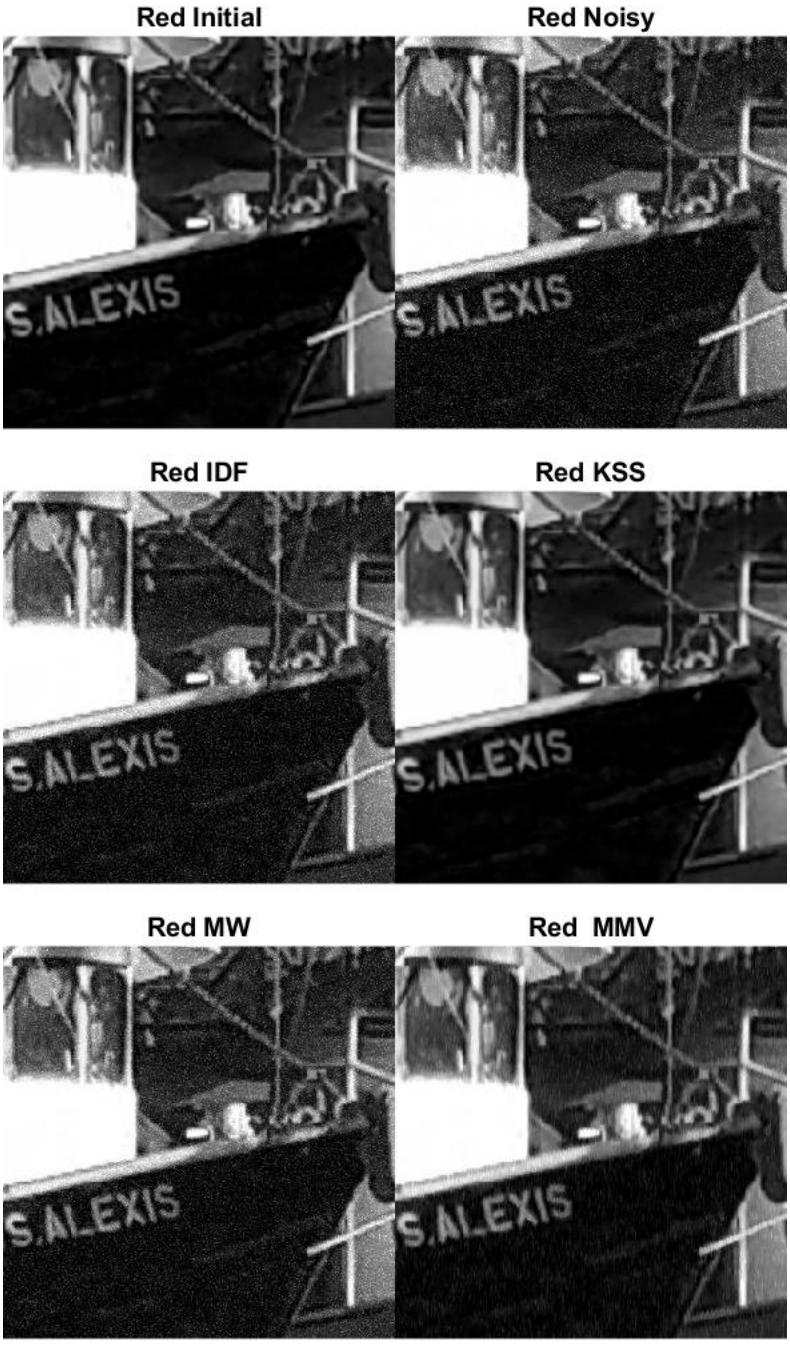

Figure 4.49: Boat Red Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

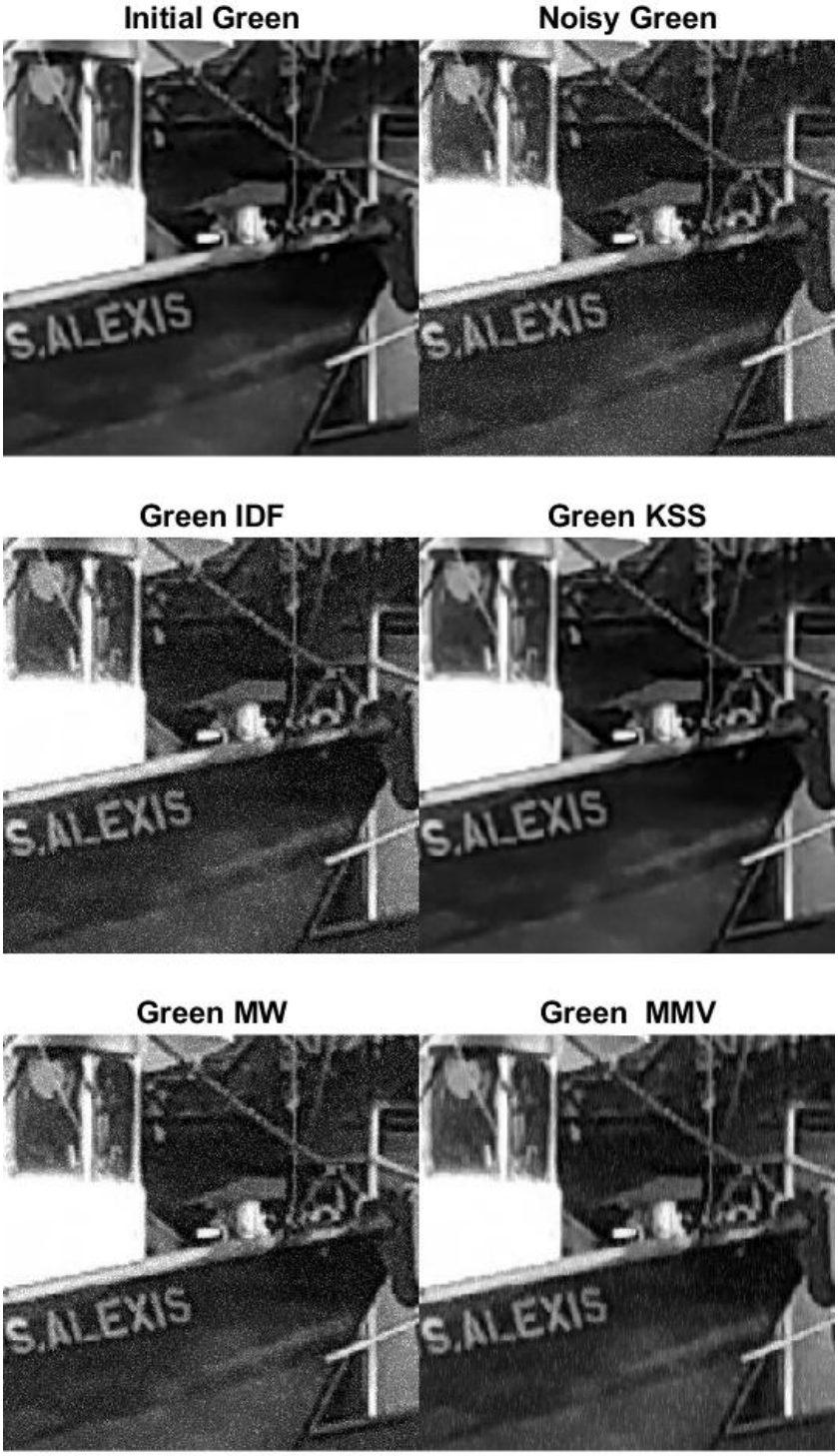

Figure 4.50: Boat Green Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

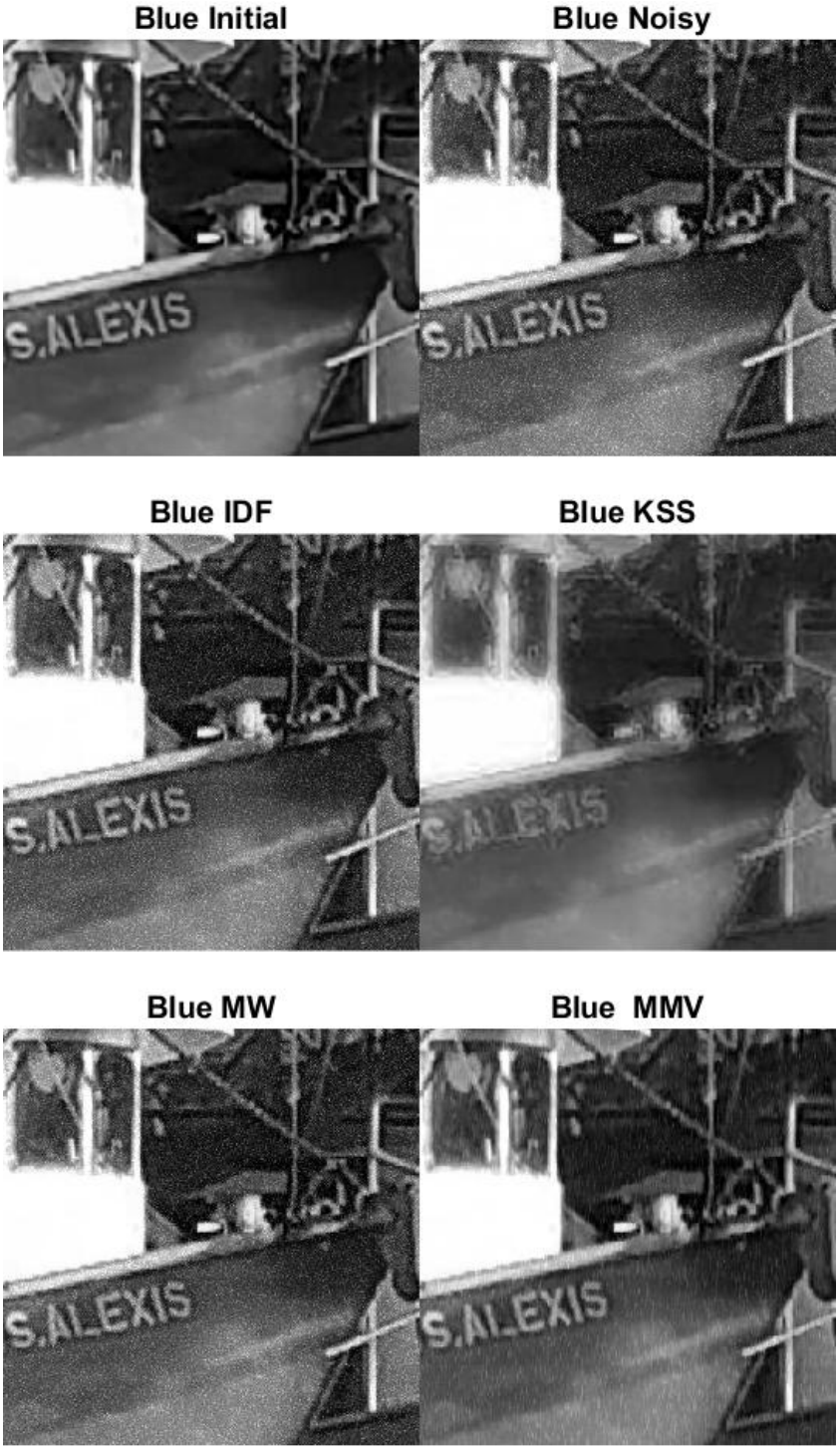

Figure 4.51: Boat Blue Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

This section contains denoised figures for the Boats test image with $n = 0.010$ noise. This image is has a challenging amount of fine line detail with many guy wires and booms on the shrimp boats. This image was created with a cell phone at a distance of 300 yards and probably has light diffraction from the water and insufficient focus. This image is a poor quality image, with more blue and less red compared to the peppers image. Some text in the lower right cropped image on the boat hull allows text character denoising.

Figures in this section show the Boat image in initial, noisy $n = 0.010$, and denoised states as whole images in Figures 4.43 to 4.47, block images in Figure 4.48 and cropped images in Figures 4.49 to 4.52. Like the Peppers image, the blue image is of poor quality, but this is not due to low blue pixel intensity magnitude, instead it is due poor quality initial blue image itself. Figure 4.52 shows the denoising process begins with a blurry blue image as seen in the initial image at the top left, and also shows all denoising methods remove of blur and sharpen the image. KSS is less successful at removing blur, but more successful at removing noise than the other 3 competitors. Overall the denoising and even deblurring is successful as demonstrated by viewing the denoised the RGB cropped imags in Figure 4.43. Even though the blue denoied image still has poor quality , this is not detectable in the final recombined color image. The text of the boat name on the hull is sharpened as seen in Figures 4.48 to 4.51, in the Red and Green images all denoising methods sharpen the text. In Figure 4.52 the blue image IDF, MMV and MW Methods do a better job in sharpening the text than the KSS method. Considering this is an image with a poor quality blue matrix, for the denoising methods to be able to sharpen the text for a image taken 300 yards away is impressive. As with prior tests examining the RGB cropped image in Figure 4.49 shows the issue with the blue blur in the KSS denoised image do not detract from the final recombined RGB image, while the lingering noise in competitor methods IDF, MMV and MW do detract, thus KSS is the best denoising method for this test.

4.2.6 Boat Shrimp Boats Test Results Noise 0.0044

Figure 4.52: Boat Whole Initial (top) Noisy $n = 0.0044$ (middle) and KSS Denoised (bottom) RGB images

Figure 4.53: Boat Whole Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.0044 noise

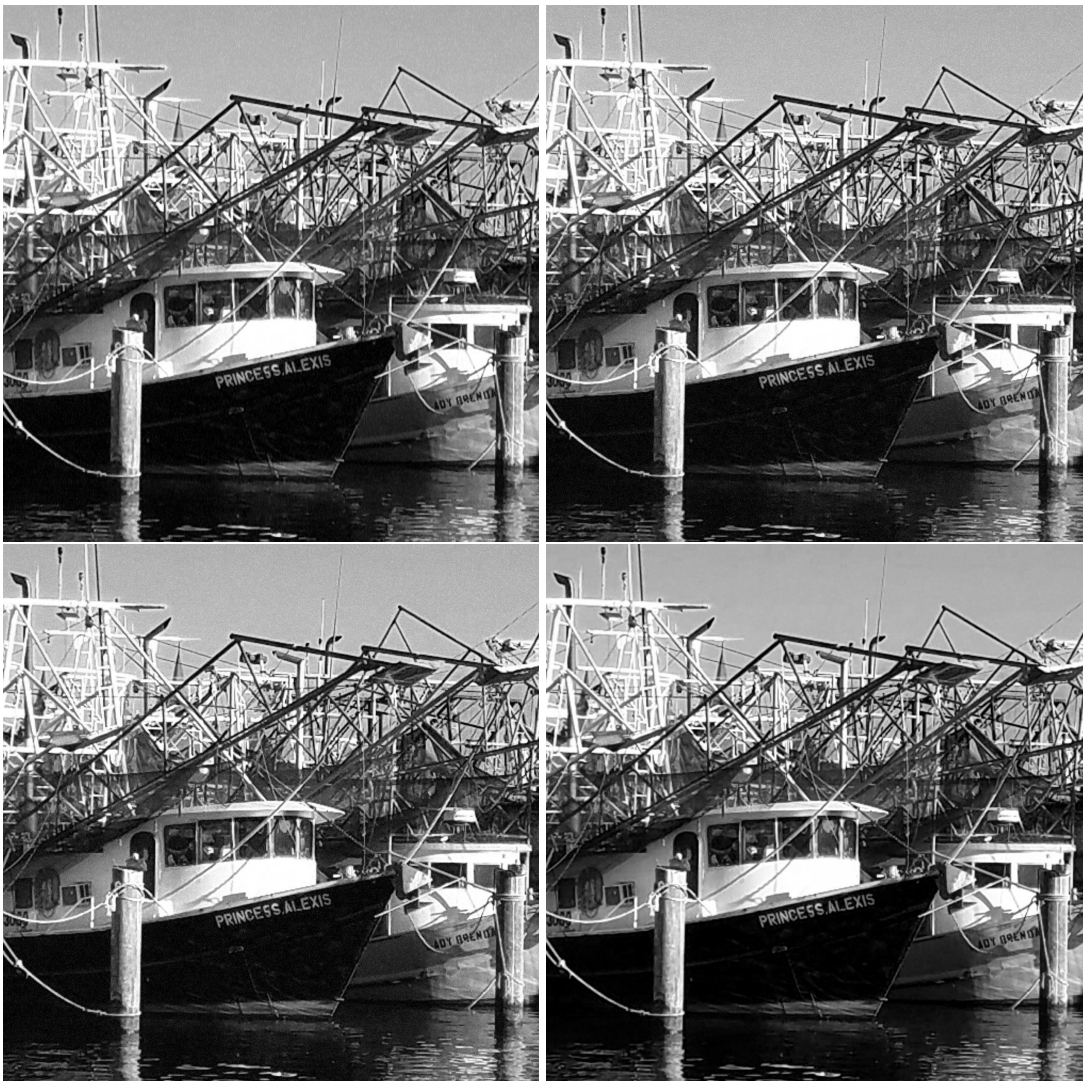

Figure 4.54: Boat Whole Red Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.55: Boat Whole Green Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.56: Boat Whole Blue Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

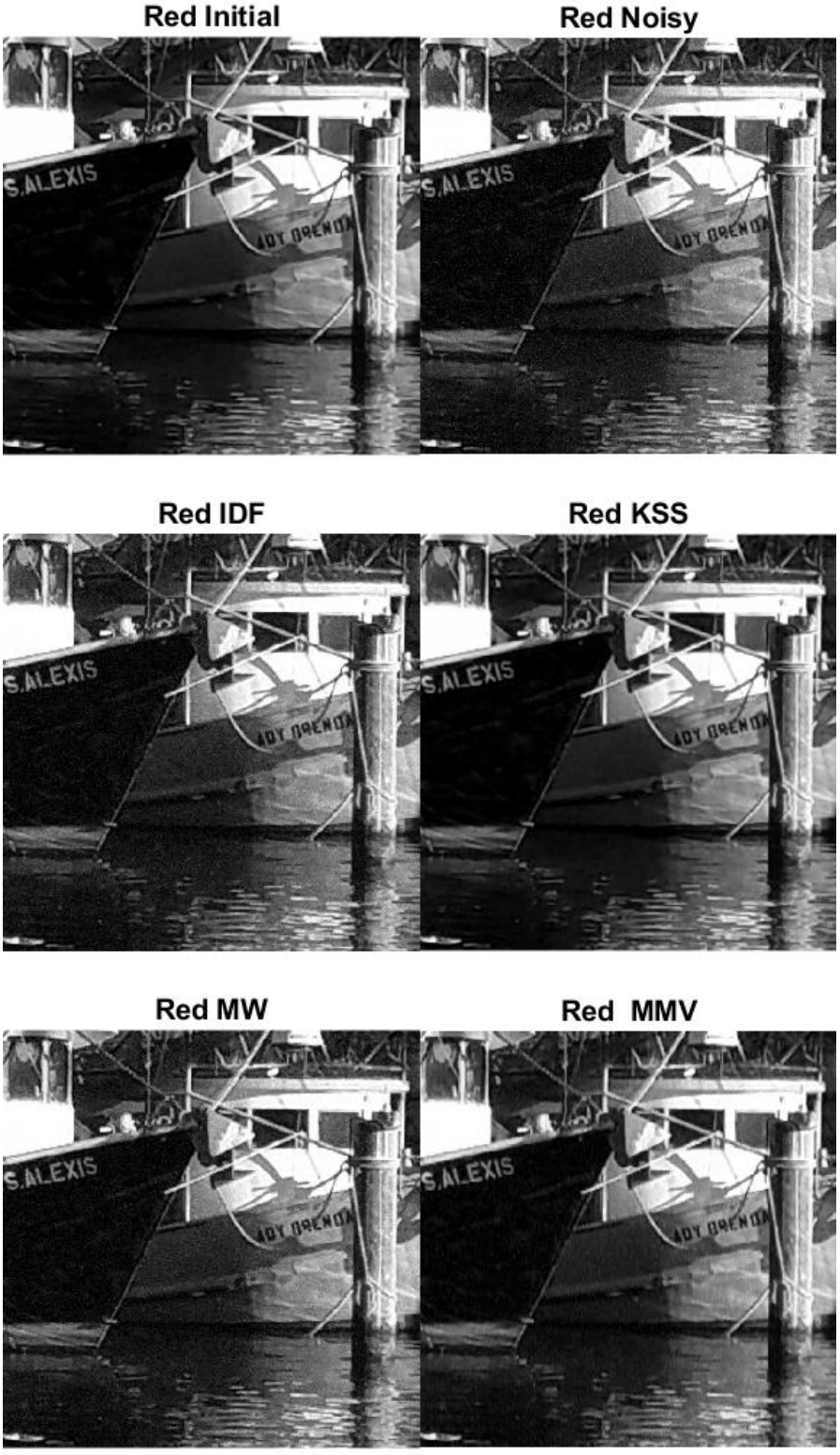

Figure 4.57: Boats Red Block 22 Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

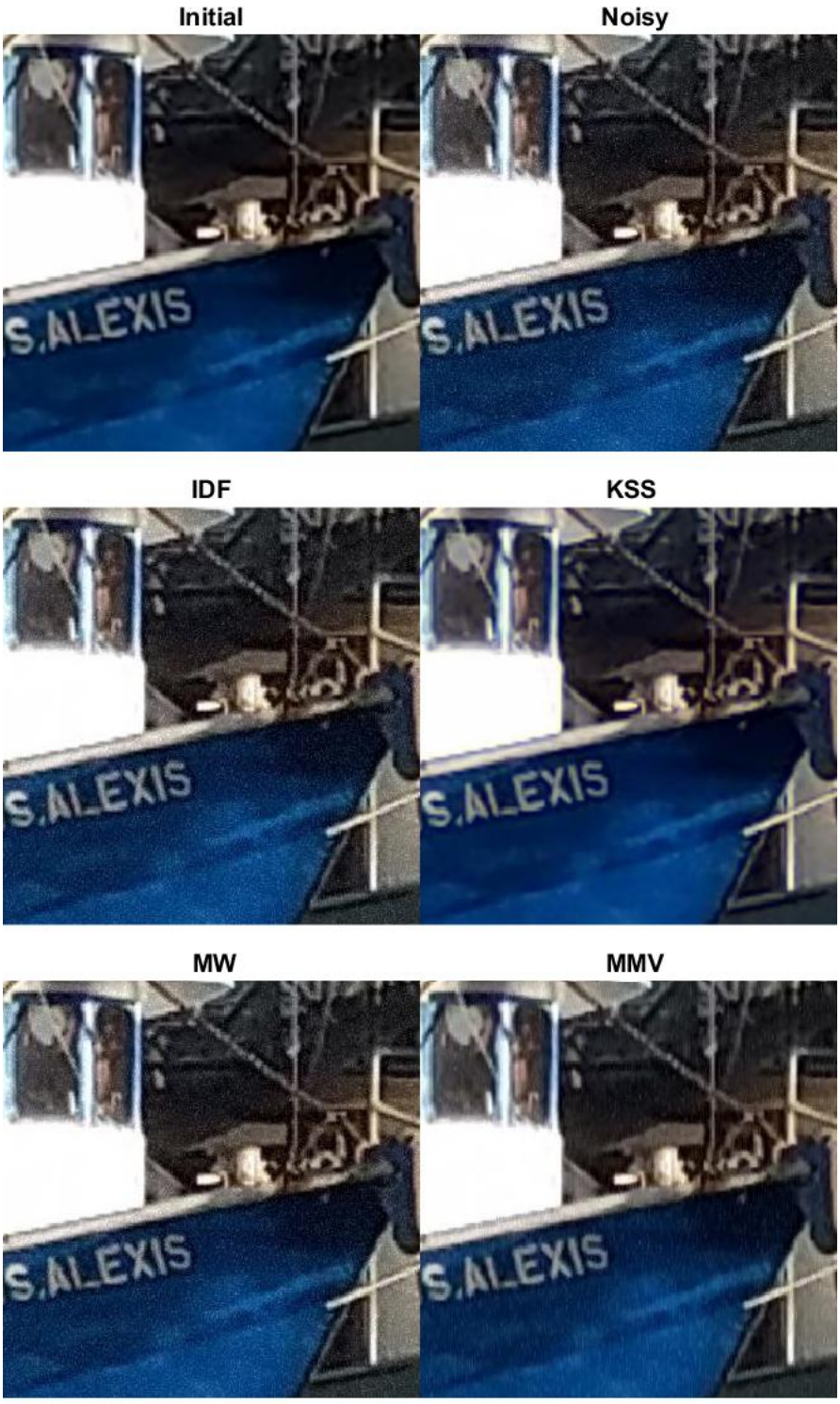

Figure 4.58: Boat RGB Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

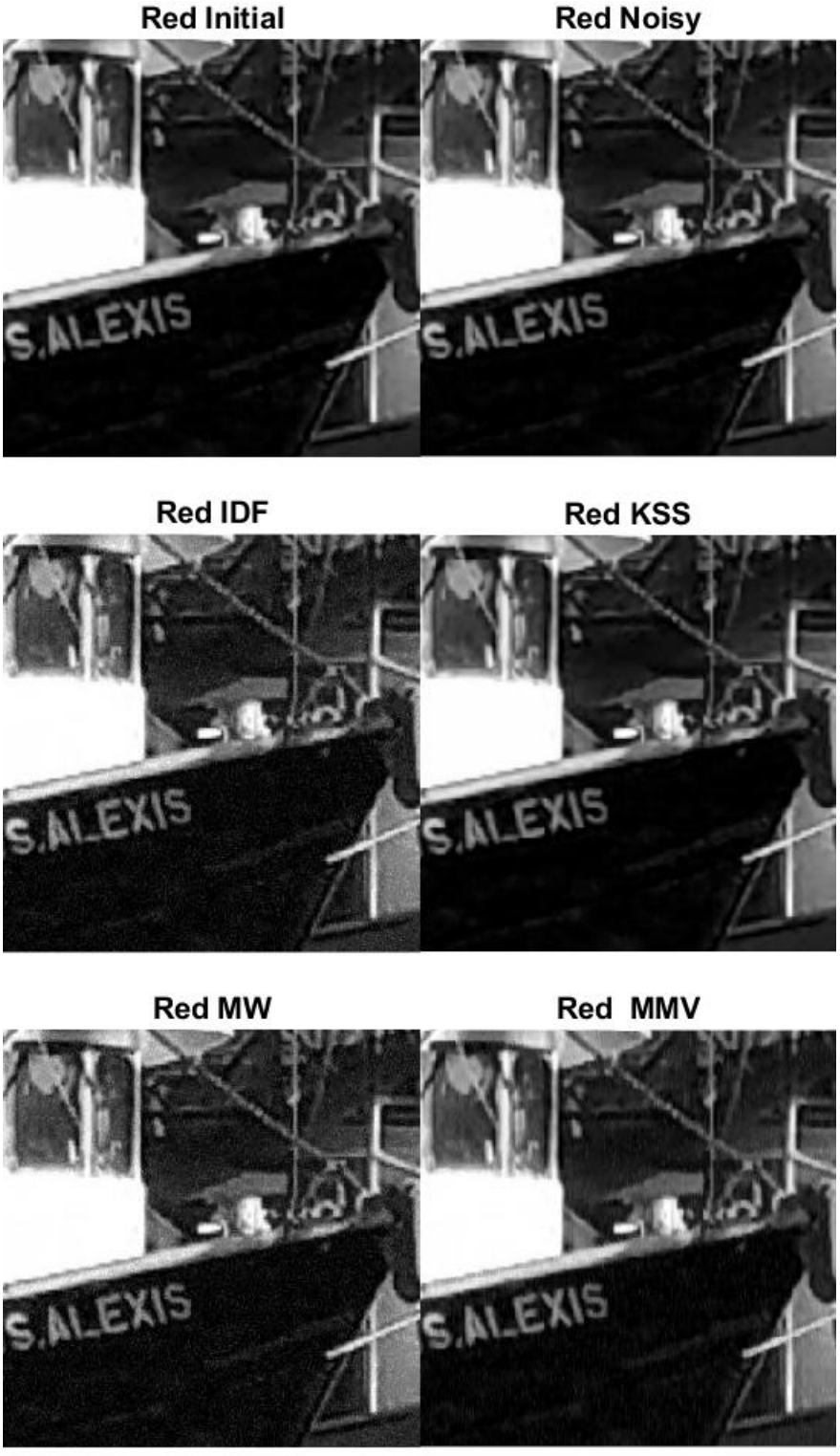

Figure 4.59: Boat Red Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

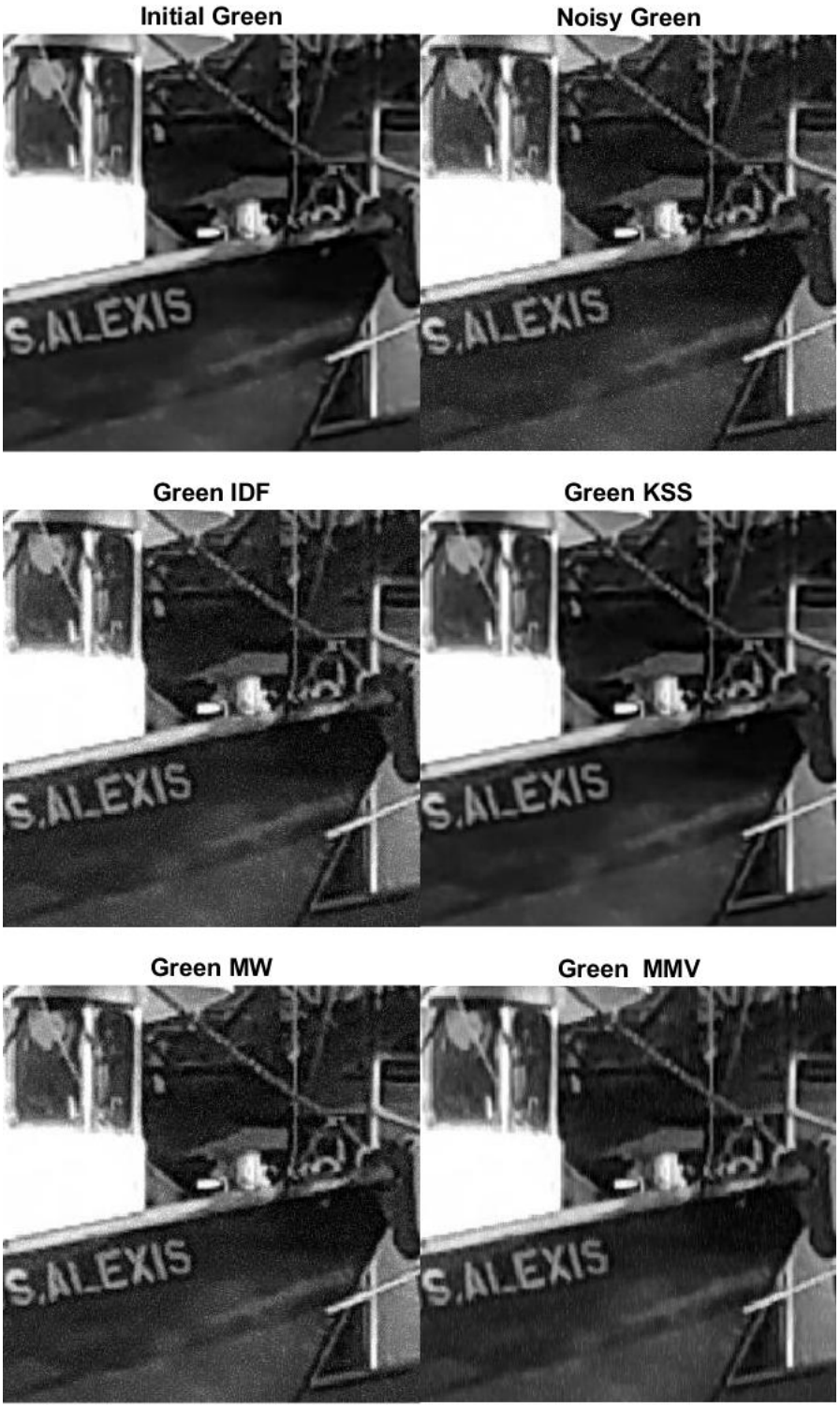

Figure 4.60: Boats Green Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

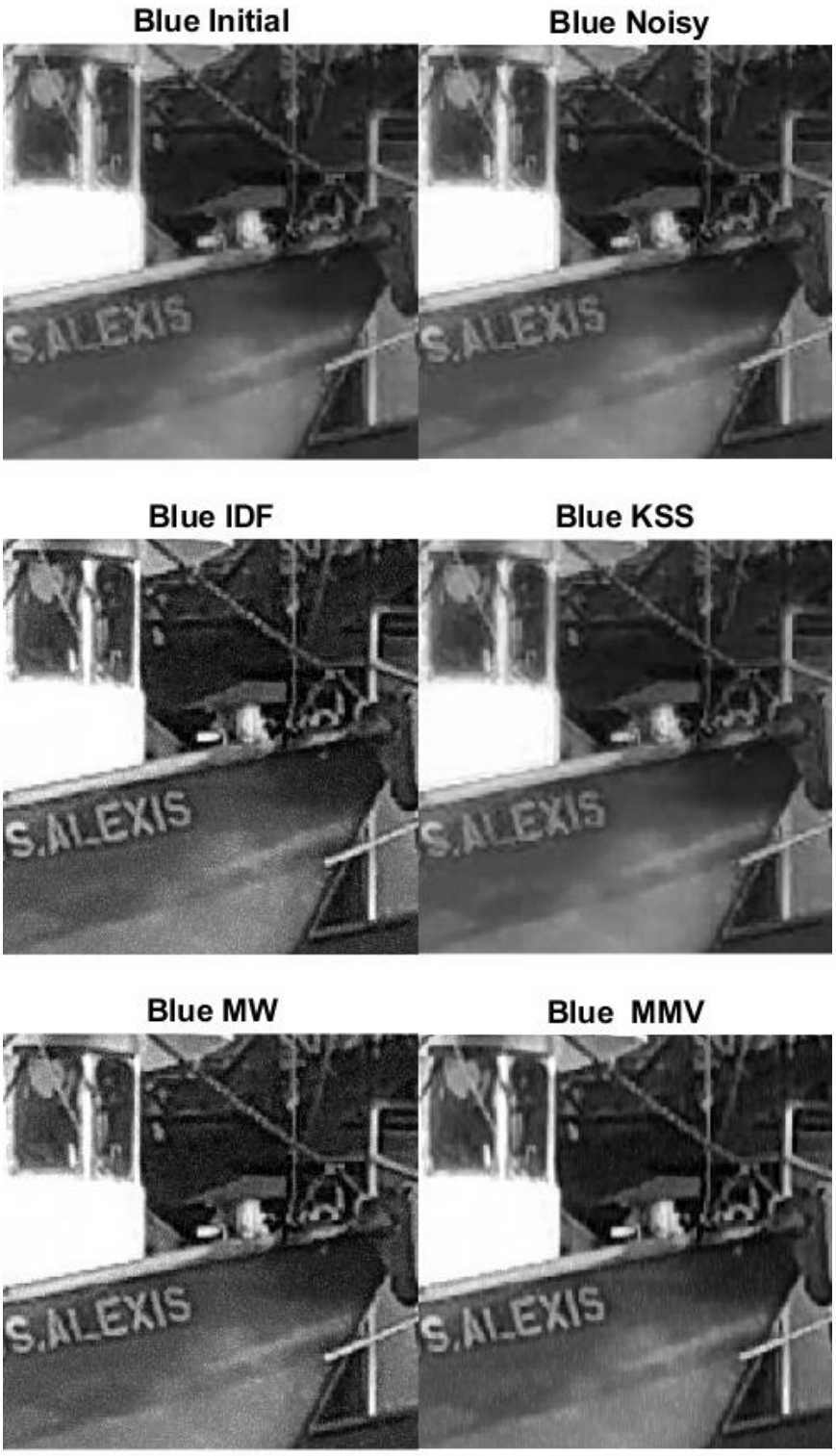

Figure 4.61: Boat Blue Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

Figures in this section show the Boats image in initial, noisy $n = 0.0044$, and denoised states as whole images in Figures 4.53 to 4.56, block images in Figure 4.58 and cropped images in Figures 4.59 to 4.62. The blue image is less blurry with this lower noise value in the initial image as is clearly seen in the cropped size Figure 4.52. At this noise level even the blue image is of a better quality and is much less blurry even in the initial image as is clearly seen in the cropped size Figure 4.52. All the denoising images reduce blur and remove noise but KSS does a better job at denoising once again, as IDF, MMV and MW have a small amount of remaining noise. Text sharpening occurs with all methods, KSS is not as good as the other methods in the blue image. Examining the recombined RGB cropped image in Figure 4.58 shows the blue blur issues with KSS dont detract from the final RGB quality, yet the lingering noise in the other methods IDF, MMV and MW does detract from final RGB quality. The KSS denoising is the best method in this test.

4.2.7 USM Test Results Noise 0.010

Figure 4.62: USM Whole Initial (top) Noisy $n = 0.010$ (middle) and KSS Denoised (bottom) RGB images

Figure 4.63: USM Whole Denoised Images: RGB (top left), Red (top right), Green (bottom left), Blue (bottom right) initially with 0.010 noise

Figure 4.64: USM Whole Red Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.65: USM Whole Green Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.66: USM Whole Blue Images: Initial and Noisy $n = 0.010$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

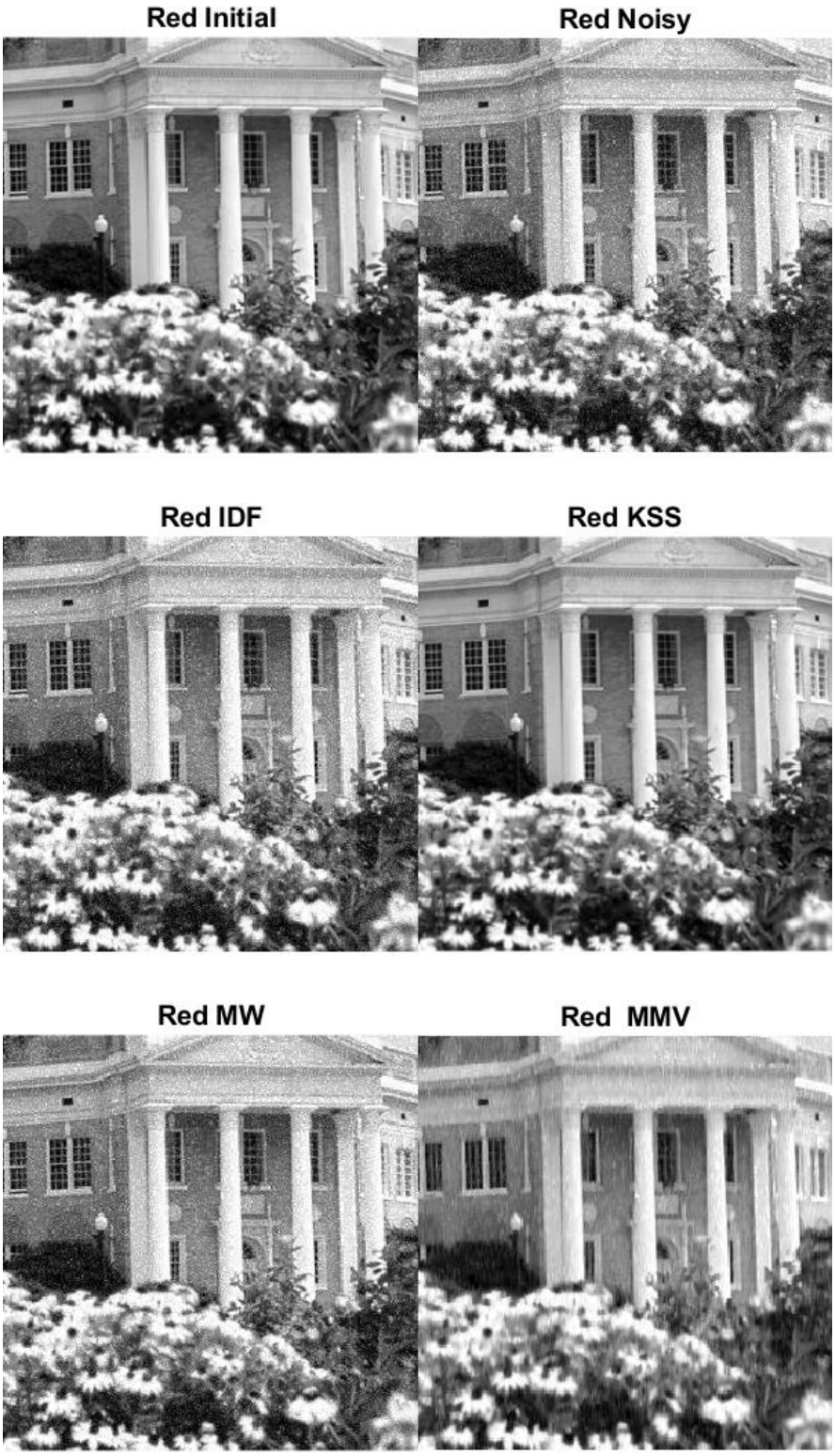

Figure 4.67: USM Red Block 22 Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

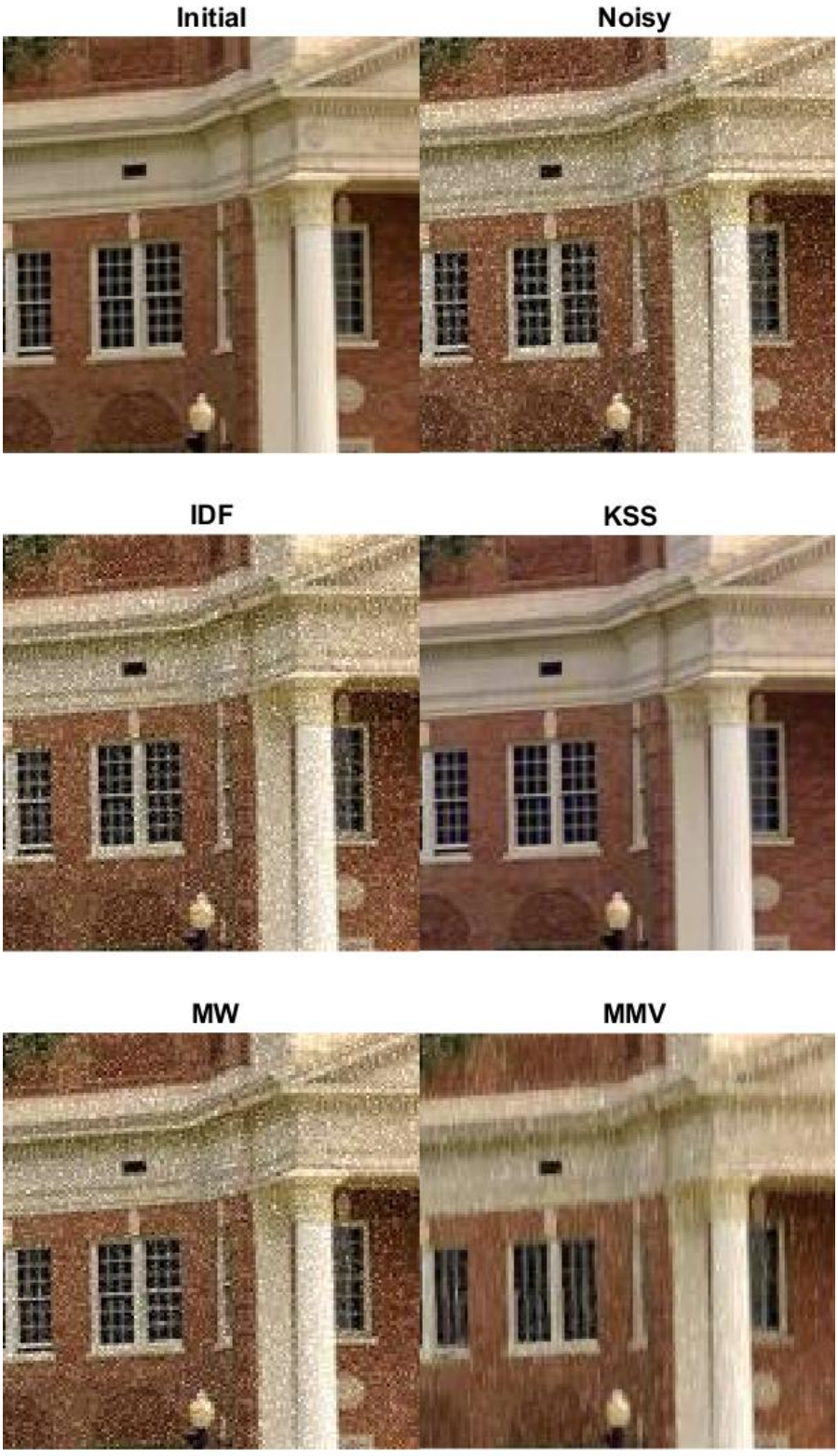

Figure 4.68: USM RGB Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

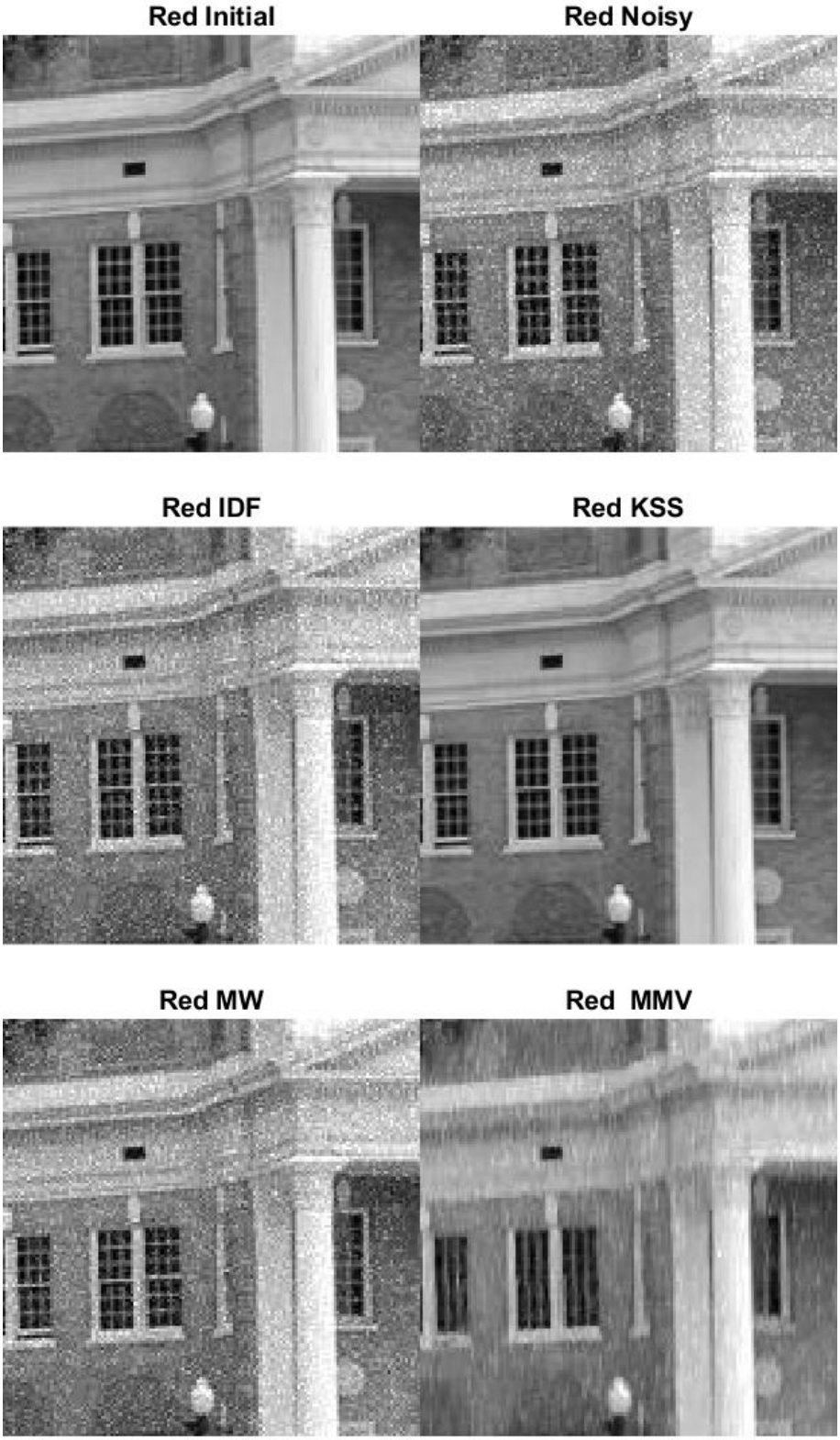

Figure 4.69: USM Red Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

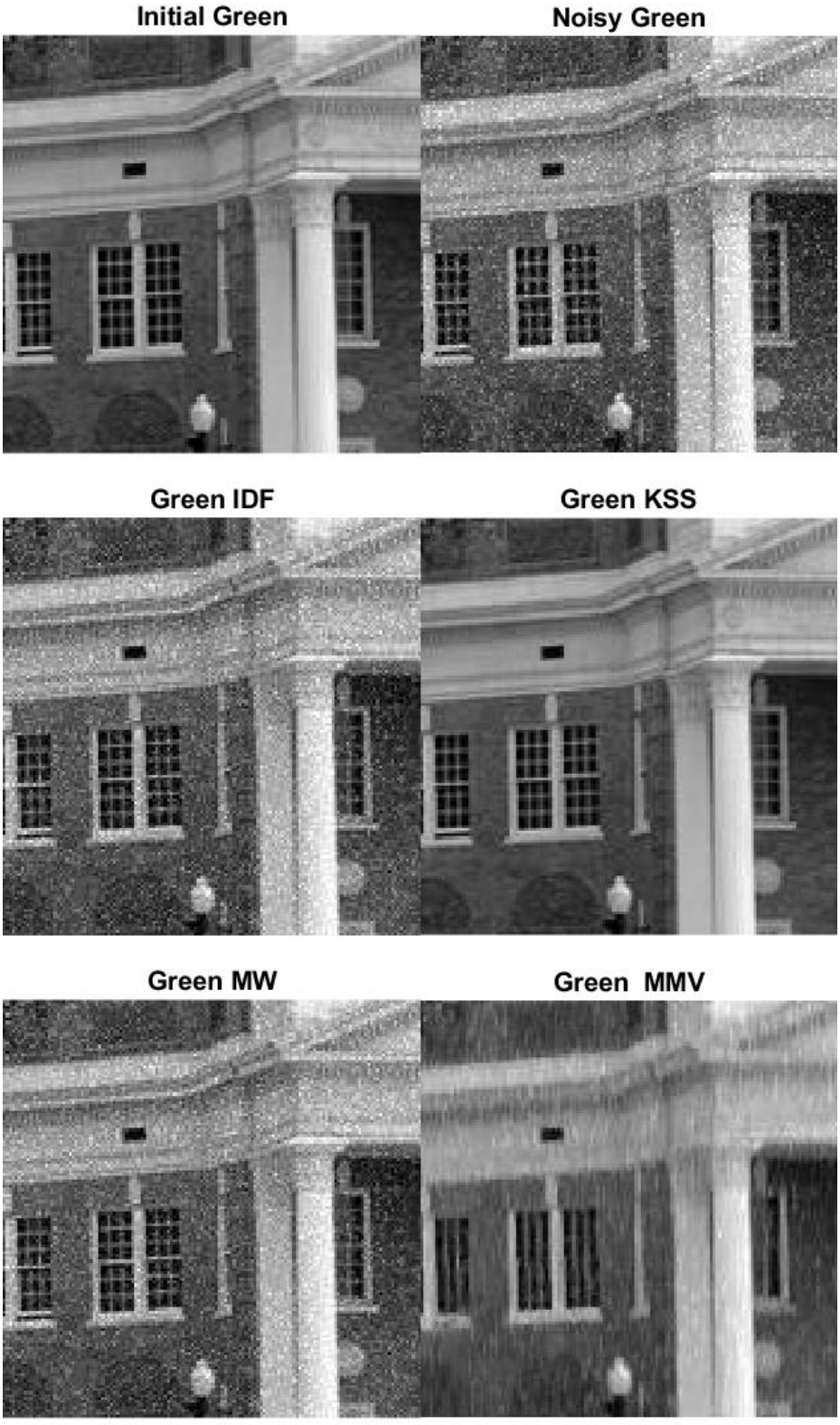

Figure 4.70: USM Green Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

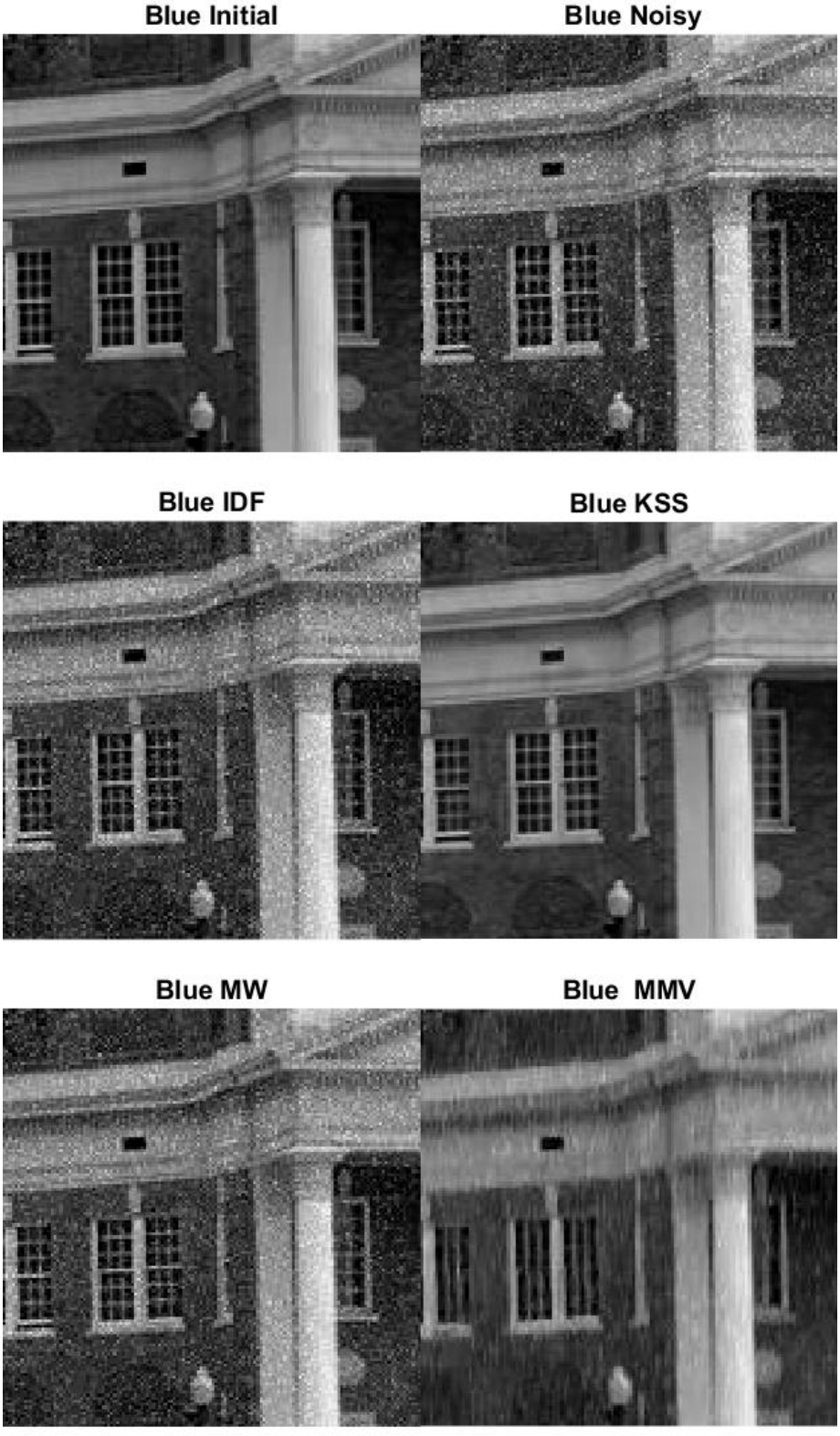

Figure 4.71: USM Blue Cropped Images: Initial and Noisy $n = 0.010$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

This section shows denoising for the USM image. The image of the USM administration building contains a lot of fine detail from the window grates and other architectural elements at risk of image loss from noise added. This image is well focused to show the detail and has a lot of high intensity near white areas, which contains high pixel magnitudes for all three color matrices. Figures in this section show the USM image in initial, noisy $n = 0.010$, and denoised states as whole images in Figures 4.62 to 4.66, block images in Figure 4.67 and cropped images in Figures 4.68 to 4.71. In the block size and crop size images noise is readily detected and make the image look like it was taken on a rare snowy day. KSS does effectively remove the noise for all three color images equally well, while the comparative methods IDF, MMV and MW do a very poor job and have lingering noise and MMV has some blur. Overall KSS does an outstanding job with this test image and the final denoised image is crisp and sharp. The superior denoising for all color images for this test image this researcher hypothesizes to be due to the image having near white regions which result in high pixel intensity in all blocks. From other tests issues with blur seemed to result when regions had either low magnitudes of a particular color, or a poor quality initial image.

4.2.8 USM Test Results Noise 0.0044

Figure 4.72: USM Whole Initial (top) Noisy $n = 0.0044$ (middle) and KSS Denoised (bottom) RGB images

Figure 4.73: USM Denoised Whole Images RGB (top left), Red (top right), Green (bottom left), and Blue (bottom right) with Noise 0.0044

Figure 4.74: USM Whole Red Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.75: USM Whole Green Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

Figure 4.76: USM Whole Blue Images: Initial and Noisy $n = 0.0044$ (top left and right), and Denoised by IDF and KSS, (middle left and right), and MMV and MW (bottom left and right)

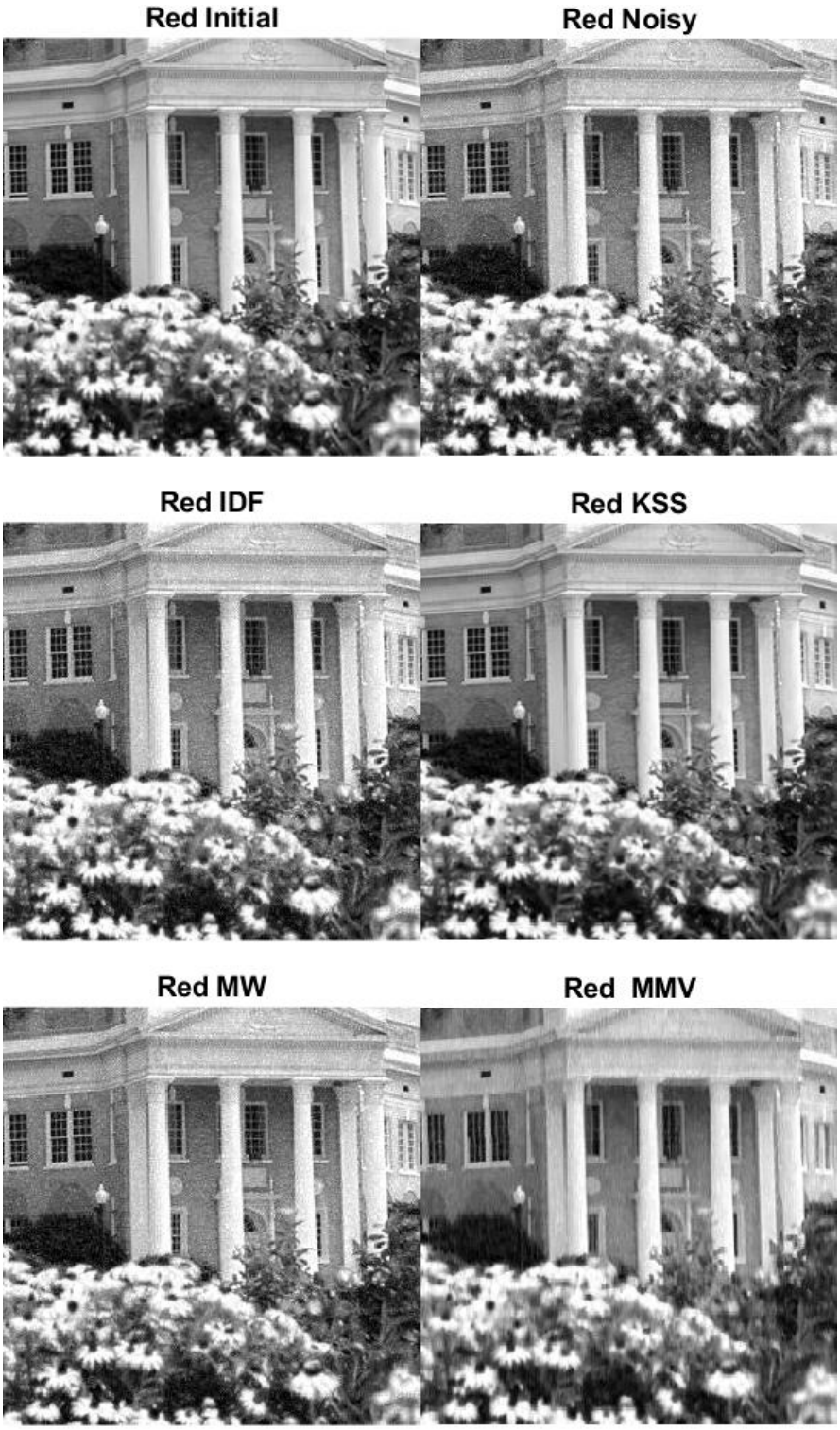

Figure 4.77: USM Red Block 22 Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

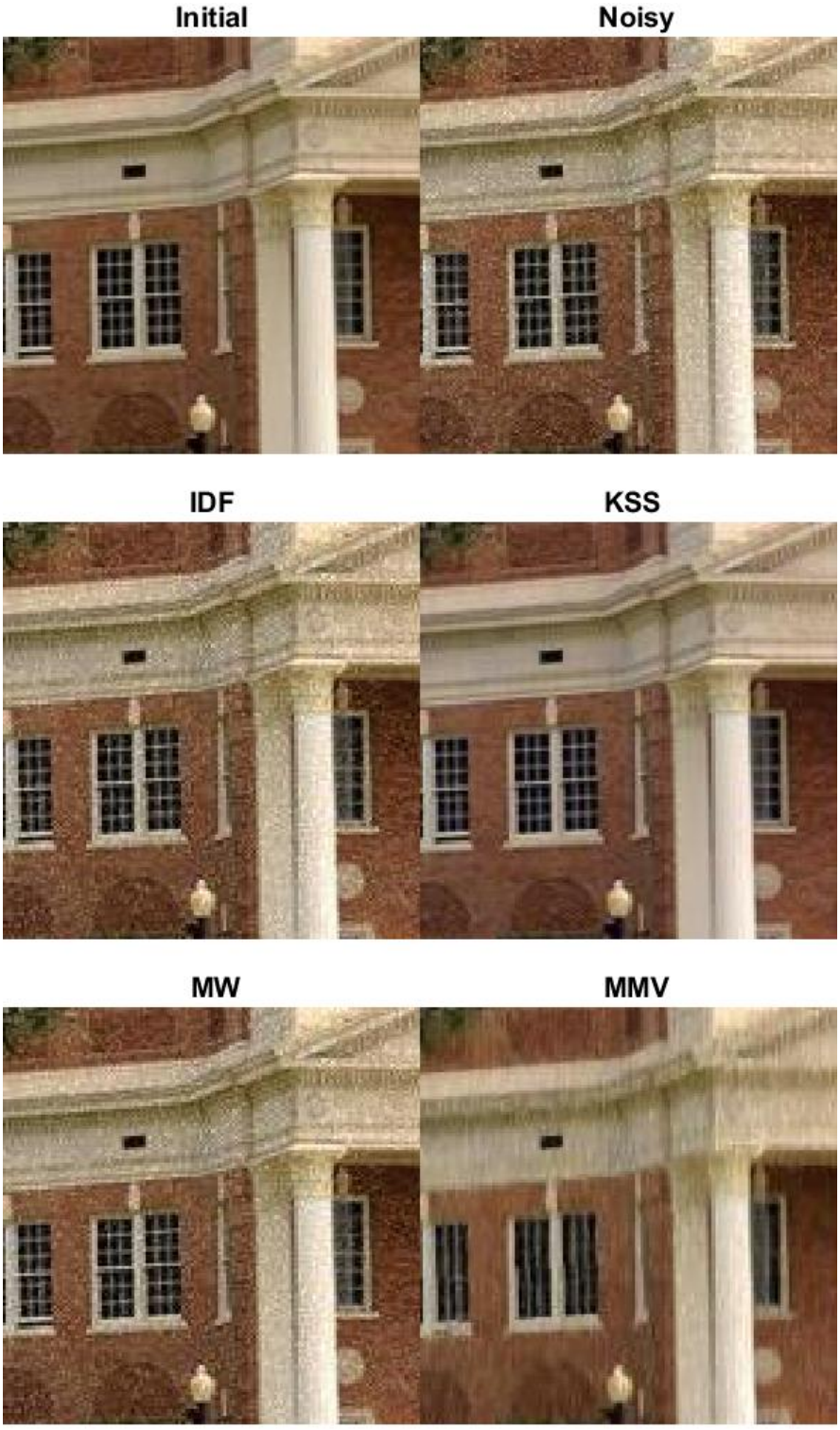

Figure 4.78: USM RGB Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

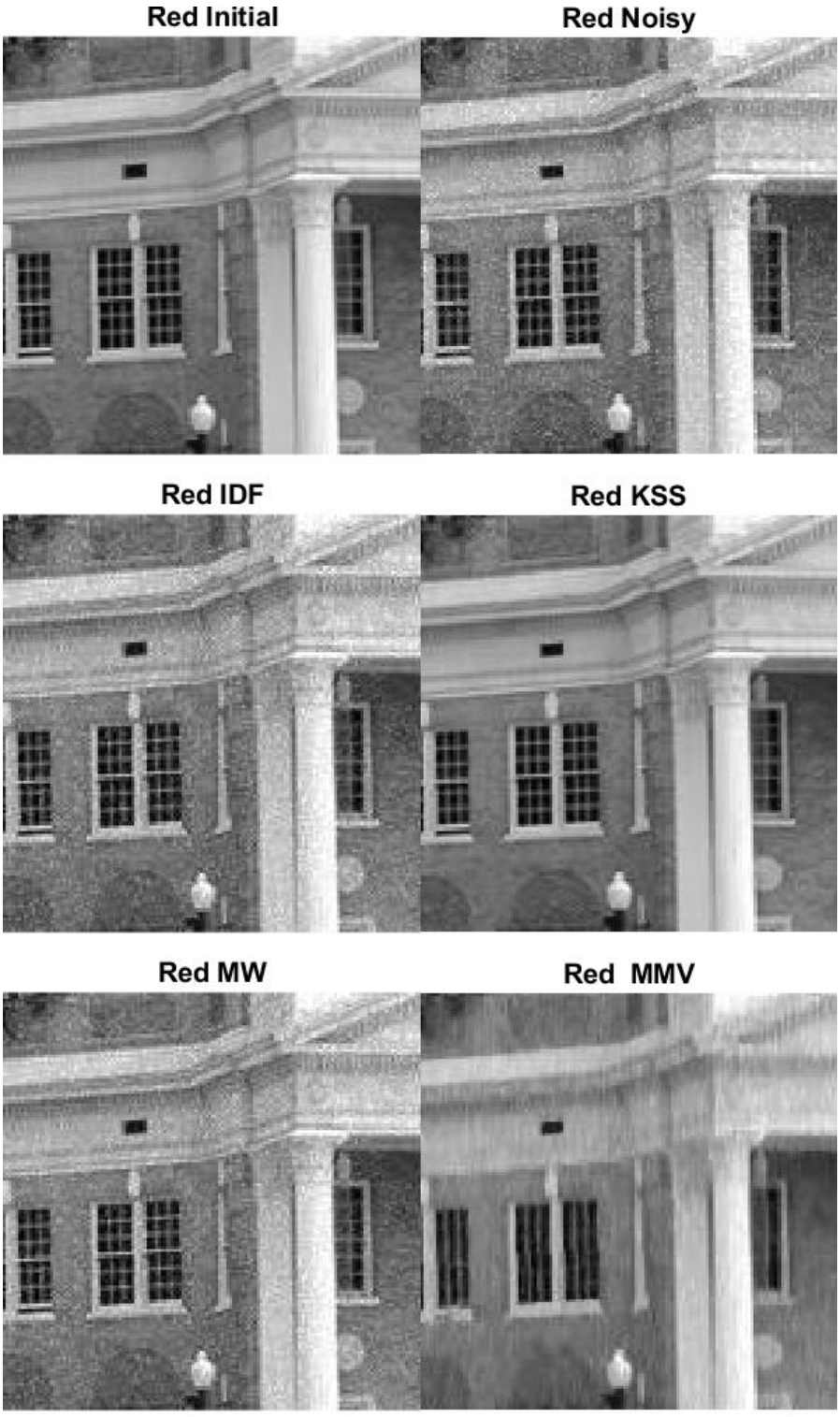

Figure 4.79: USM Red Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

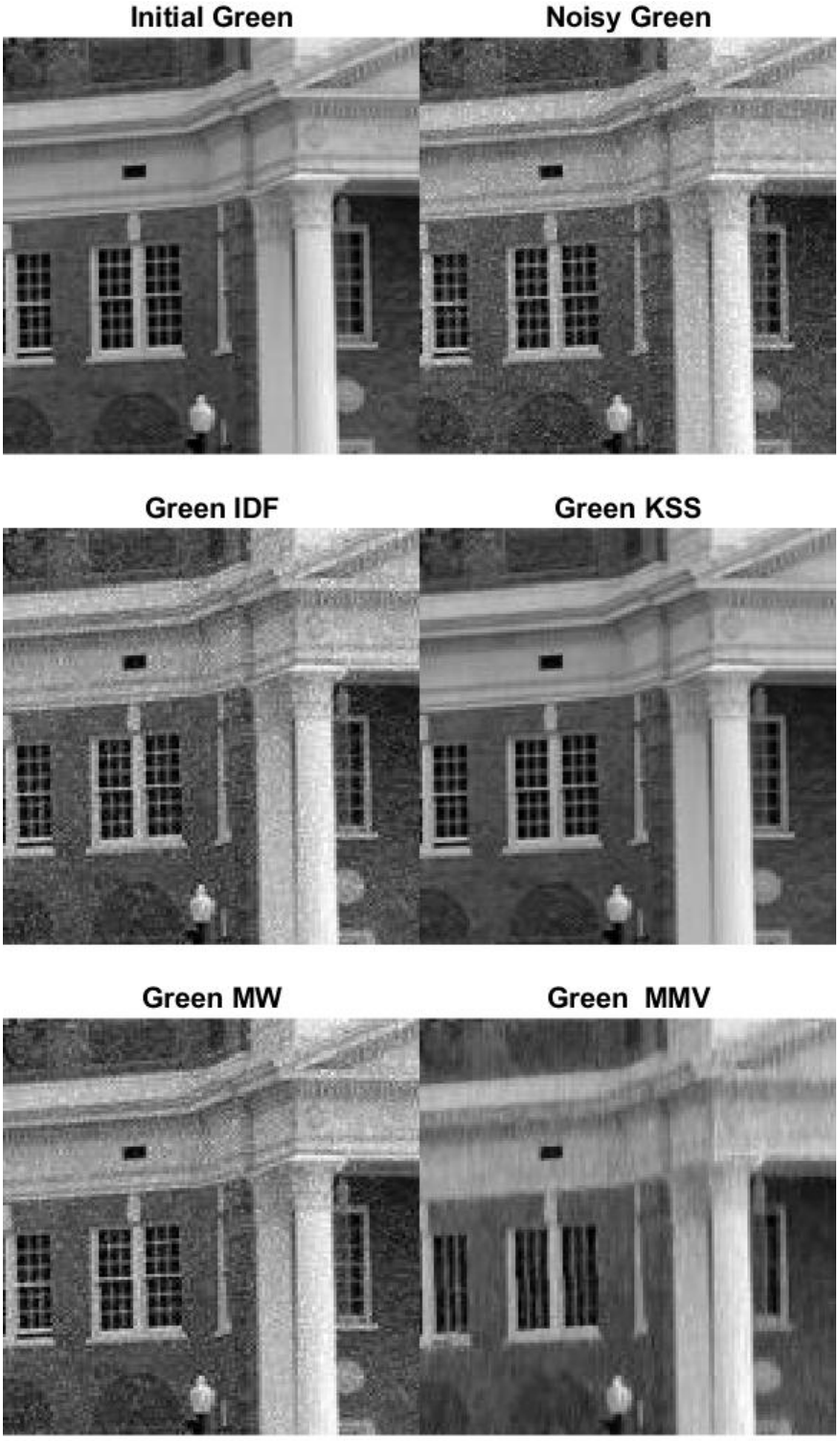

Figure 4.80: USM Green Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

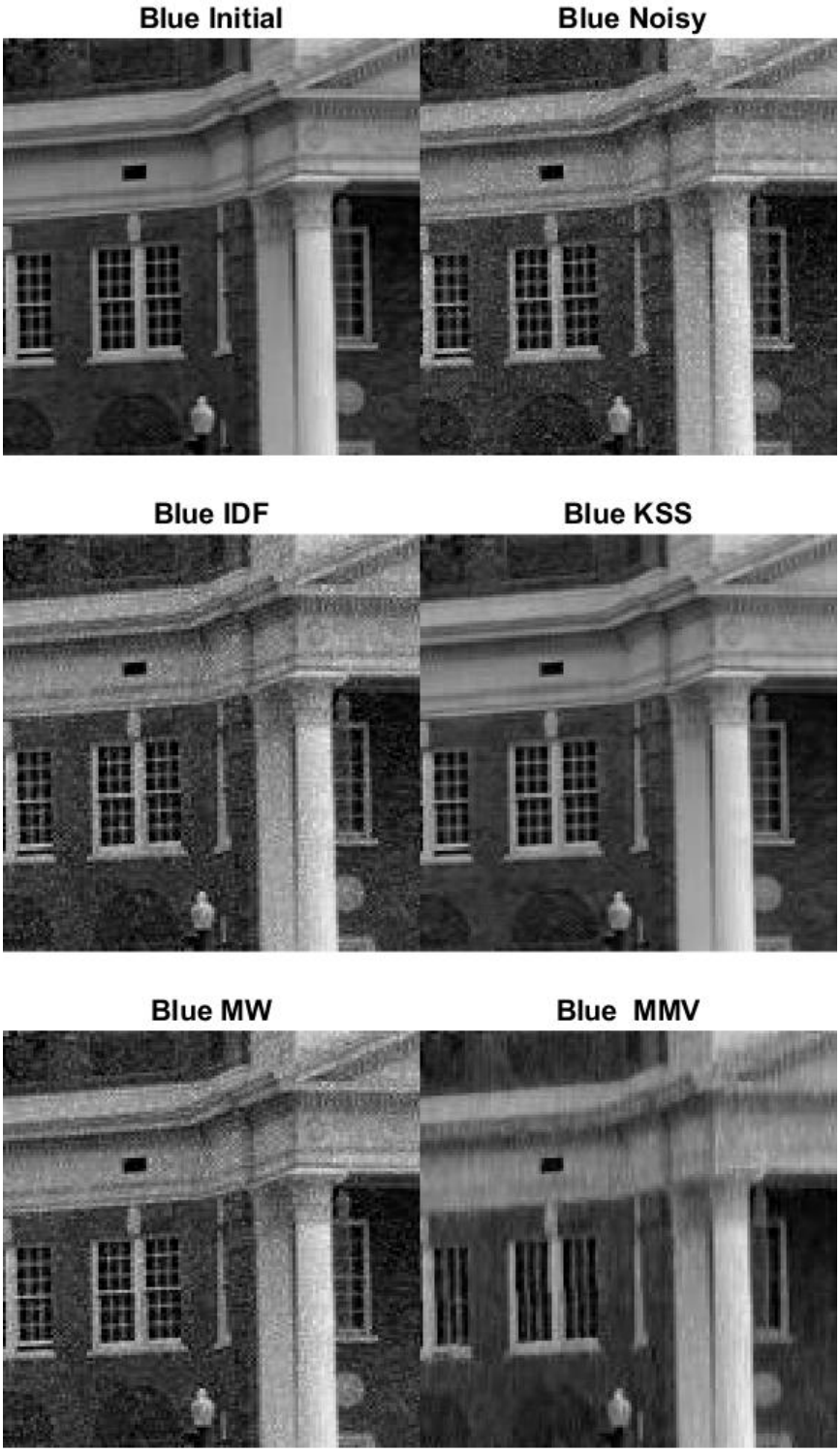

Figure 4.81: USM Blue Cropped Images: Initial and Noisy $n = 0.0044$ (top left and right), Denoised results from Methods IDF and KSS, (middle left and right), MMV, MW (bottom left and right)

Figures in this section show the USM image in initial, noisy $n = 0.0044$, and denoised states as whole images in Figures 4.73 to 4.77, block images in Figure 4.78 and cropped images in Figures 4.79 to 4.82. The results for this test follow the the higher noise test but reach even better final quality due to the reduced noise. KSS does effectively remove the noise for all three color images equally well, while the comparative methods IDF, MMV and MW do a very poor job, though not as bad as the higher noise test, and have lingering noise and MMV has some blur. Once again KSS does an outstanding job with this test image and the final denoised image is crisp and sharp. The superior denoising for all color images for this test image it is hypothesized to be due to the image having near white regions which result in high pixel intensity in all blocks. From prior tests issues with blur seemed to result when regions had either low magnitudes of a particular color, or a poor quality initial image. An examination of Figure 4.79 the RGB cropped images show KSS to be the superior denoising method for this test.

4.2.9 Tabular Results Parameter Optimization Each Test

Table 4.1 provides a summary of abbreviated terms that appear in the tables included in this section. Tables 4.2 to 4.9 summarize the parameter optimization for each block of each test image, note how only the dt parameter varies between blocks while the k and ε variables are held the same as those optimized for the first red block. Table 4.18 summarizes the parameter optimization for all R11 blocks of each test.

Table 4.2: Key to Abbreviations in Results Tables

Abbreviation	Description
Cl	Color of image begin tested
R	Red , the Color of image reduced to a single character
G	Green , the Color of image reduced to a single character
B	Blue , the Color of image reduced to a single character
Im	Image begin tested
Land or L	Landsat image name shortened to 4 or 1 characters
Pepp or P	Pepper image name shortened to 4 or 1 characters
B	Boat image name shortened to single character
U	USM image name shortened to single character
tb	best time in seconds
dtb	best time step in seconds
kb	best k Threshold parameter
eps	abbreviation of ε regularizing parameter
epsb	best ε regularizing parameter
mse0	initial MSE
PSNR0	initial $PSNR$
Ssim0	initial $Ssim$
tott	total parameter optimization time in seconds
TQ0	total quality TQ of noisy initial image
(114r)	version of authors **MATLAB** code used
nb	index of best T loop iteration
tb*5	best time $t \times 10^{-5}$ seconds
dtb*5	best timestep $dt \times 10^{-5}$ seconds
epsb	best ε regularizing parameter
kb*3	best edge detection parameter $ktimes10^{-3}$
TQb	best TQ total quality
pTQ	percent change in decimal form for TQ
ub	usebests flag if if 0 parameter optimization is on
pPSNR	percent change in decimal form for $PSNR$
σ^2*3	Gaussian Noise variance $\sigma^2 \times 10^3$
MTHsh	MTHs and MTHh in Table when they are equal

Parameter Optimization R11 Landsat $n = 0.010$

Table 4.3: Parameter Optimization Block R11 Landsat = 0.010 (114r) 3-23-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	3	0.390625	0.097656	0.008	3.1250	35.1388	25	28.2006	0
R12	3	0.390625	0.097656	0.008	3.1250	35.6126	26	28.2166	1
R21	3	0.390625	0.097656	0.008	3.1250	35.4694	26	28.2198	1
R22	3	0.390625	0.097656	0.008	3.1250	34.8446	24	28.1902	1
G11	3	0.390625	0.097656	0.008	3.1250	34.5512	23	28.1728	1
G12	3	0.390625	0.097656	0.008	3.1250	34.9046	24	28.1794	1
G21	3	0.390625	0.097656	0.008	3.1250	34.9991	24	28.1932	1
G22	4	0.488281	0.097656	0.008	3.1250	34.4656	22	28.1747	1
B11	4	0.488281	0.097656	0.008	3.1250	31.1785	11	28.0888	1
B12	3	0.390625	0.097656	0.008	3.1250	32.8997	17	28.1228	1
B21	3	0.390625	0.097656	0.008	3.1250	31.8435	13	28.1077	1
B22	4	0.488281	0.097656	0.008	3.1250	31.5328	12	28.0878	1

Parameter Optimization R11 Landsat $n = 0.0044$

Table 4.4: Parameter Optimization Block R11 Landsat = 0.0044 (114r) 3-23-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	3	0.781250	0.195313	0.008	12.5000	37.8069	29	29.4187	0
R12	3	0.781250	0.195313	0.008	12.5000	38.2904	30	29.4321	1
R21	4	0.976563	0.195313	0.008	12.5000	38.1994	30	29.4335	1
R22	4	0.976563	0.195313	0.008	12.5000	37.5407	28	29.4090	1
G11	4	0.976563	0.195313	0.008	12.5000	37.3171	27	29.3956	1
G12	4	0.976563	0.195313	0.008	12.5000	37.7136	28	29.4031	1
G21	4	0.976563	0.195313	0.008	12.5000	37.7256	28	29.4108	1
G22	4	0.976563	0.195313	0.008	12.5000	37.1667	26	29.3890	1
B11	5	1.171875	0.195313	0.008	12.5000	34.0441	16	29.3068	1
B12	4	0.976563	0.195313	0.008	12.5000	36.0133	23	29.3508	1
B21	4	0.976563	0.195313	0.008	12.5000	34.8719	19	29.3288	1
B22	5	1.171875	0.195313	0.008	12.5000	34.3626	17	29.3054	1

Parameter Optimization R11 Block Peppers $n = 0.010$

Table 4.5: Parameter Optimization Block R11 Peppers = 0.010 (114r) 3-24-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	2	1.171875	0.390625	0.008	3.1250	30.3133	10	27.4978	0
R12	2	0.073242	0.024414	0.008	3.1250	32.4064	17	27.7839	1
R21	2	0.146484	0.048828	0.008	3.1250	34.4479	23	27.9992	1
R22	2	0.073242	0.024414	0.008	3.1250	34.0397	22	27.8248	1
G11	2	1.562500	0.781250	0.008	3.1250	27.0351	-2	27.4705	1
G12	2	1.171875	0.390625	0.008	3.1250	31.5119	14	27.5288	1
G21	2	1.171875	0.390625	0.008	3.1250	32.9919	20	27.5533	1
G22	2	1.171875	0.390625	0.008	3.1250	32.5389	18	27.5198	1
B11	2	1.171875	0.390625	0.008	3.1250	29.3348	7	27.4763	1
B12	3	0.781250	0.195313	0.008	3.1250	30.7840	12	27.5137	1
B21	2	1.171875	0.390625	0.008	3.1250	31.7340	15	27.5399	1
B22	2	1.171875	0.390625	0.008	3.1250	31.1531	13	27.5153	1

Parameter Optimization Peppers $n = 0.0044$

Table 4.6: Parameter Optimization Block R11 Peppers = 0.0044 (114r) 3-23-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	2	0.585938	0.195313	0.008	3.1250	33.0111	15	28.6952	0
R12	2	0.036621	0.012207	0.008	3.1250	34.9839	21	28.9748	1
R21	2	0.073242	0.024414	0.008	3.1250	36.8458	26	29.1535	1
R22	2	0.036621	0.012207	0.008	3.1250	36.5207	26	29.0181	1
G11	2	0.585938	0.195313	0.008	3.1250	32.0103	12	28.6682	1
G12	2	0.585938	0.195313	0.008	3.1250	33.9026	18	28.7233	1
G21	2	0.585938	0.195313	0.008	3.1250	35.4323	23	28.7569	1
G22	2	0.585938	0.195313	0.008	3.1250	35.0690	22	28.7277	1
B11	2	0.585938	0.195313	0.008	3.1250	32.3537	13	28.6840	1
B12	2	0.585938	0.195313	0.008	3.1250	33.2523	16	28.7242	1
B21	2	0.585938	0.195313	0.008	3.1250	34.4360	20	28.7517	1
B22	2	0.585938	0.195313	0.008	3.1250	33.9367	18	28.7260	1

Parameter Optimization R11 Boat $n = 0.010$

Table 4.7: Parameter Optimization Block R11 Boat = 0.010 (114r) 3-30-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	2	0.146484	0.048828	0.008	3.1250	35.6941	18	30.3014	0
R12	4	0.244141	0.048828	0.008	3.1250	33.4437	11	30.0379	1
R21	2	0.146484	0.048828	0.008	3.1250	38.9205	28	30.2987	1
R22	2	0.018311	0.006104	0.008	3.1250	39.4130	30	30.4067	1
G11	2	0.146484	0.048828	0.008	3.1250	35.7160	17	30.4463	1
G12	4	0.244141	0.048828	0.008	3.1250	33.6504	12	30.1698	1
G21	2	0.146484	0.048828	0.008	3.1250	39.0330	29	30.2842	1
G22	3	0.097656	0.024414	0.008	3.1250	39.3776	30	30.3893	1
B11	2	0.146484	0.048828	0.008	3.1250	36.7304	19	30.9068	1
B12	3	0.195313	0.048828	0.008	3.1250	34.6286	13	30.7157	1
B21	3	0.195313	0.048828	0.008	3.1250	39.1629	30	30.2004	1
B22	2	0.146484	0.048828	0.008	3.1250	39.2985	29	30.3486	1

Parameter Optimization R11 Boat $n = 0.0044$

Table 4.8: Parameter Optimization Block R11 Boat = 0.0044 (114r) 3-30-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	2	0.146484	0.048828	0.008	3.1250	37.5129	19	31.4580	0
R12	3	0.195313	0.048828	0.008	3.1250	35.5321	31.2163	1	
R21	2	0.073242	0.024414	0.008	3.1250	41.6361	32	31.4875	1
R22	2	0.018311	0.006104	0.008	3.1250	41.3637	31	31.5941	1
G11	2	0.073242	0.024414	0.008	3.1250	37.8169	20	31.5106	1
G12	3	0.195313	0.048828	0.008	3.1250	35.5664	14	31.2449	1
G21	2	0.073242	0.024414	0.008	3.1250	41.6813	32	31.4728	1
G22	2	0.073242	0.024414	0.008	3.1250	41.6055	32	31.5723	1
B11	2	0.146484	0.048828	0.008	3.1250	38.0383	20	31.7424	1
B12	2	0.146484	0.048828	0.008	3.1250	35.9806	14	31.5107	1
B21	2	0.146484	0.048828	0.008	3.1250	41.4778	32	31.3846	1
B22	3	0.097656	0.024414	0.008	3.1250	41.5073	32	31.5191	1

Parameter Optimization R11 USM $n = 0.010$

Table 4.9: Parameter Optimization Block R11 USM = 0.010 (114r) 3-25-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	3	0.195313	0.048828	0.008	3.1250	36.6106	27	28.9264	0
R12	3	0.195313	0.048828	0.008	3.1250	32.4531	9	29.6384	1
R21	3	0.195313	0.048828	0.008	3.1250	36.8668	28	28.9127	1
R22	4	0.244141	0.048828	0.008	3.1250	37.5188	29	28.9807	1
G11	3	0.195313	0.048828	0.008	3.1250	36.6858	27	28.9062	1
G12	4	0.244141	0.048828	0.008	3.1250	32.0480	9	29.4476	1
G21	4	0.244141	0.048828	0.008	3.1250	36.6608	27	28.7878	1
G22	4	0.244141	0.048828	0.008	3.1250	37.1681	29	28.8141	1
B11	3	0.195313	0.048828	0.008	3.1250	36.0041	25	28.8299	1
B12	4	0.244141	0.048828	0.008	3.1250	31.4309	8	28.9960	1
B21	4	0.244141	0.048828	0.008	3.1250	34.7269	21	28.7318	1
B22	4	0.244141	0.048828	0.008	3.1250	35.6547	24	28.7529	1

Parameter Optimization R11 USM $n = 0.0044$

Table 4.10: Parameter Optimization Block R11 USM = 0.0044 (114r) 3-25-21

Block	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
R11	4	0.488281	0.097656	0.008	12.5000	39.1680	30	30.0551	0
R12	4	0.488281	0.097656	0.008	12.5000	33.9820	11	30.5108	1
R21	5	0.585938	0.097656	0.008	12.5000	39.4009	31	30.0514	1
R22	4	0.488281	0.097656	0.008	12.5000	39.8849	33	30.0945	1
G11	3	0.390625	0.097656	0.008	12.5000	39.1944	31	30.0273	1
G12	4	0.488281	0.097656	0.008	12.5000	33.5269	11	30.3056	1
G21	5	0.585938	0.097656	0.008	12.5000	39.2347	31	29.9484	1
G22	5	0.585938	0.097656	0.008	12.5000	39.6035	32	29.9612	1
B11	4	0.488281	0.097656	0.008	12.5000	38.6186	29	29.9593	1
B12	5	0.585938	0.097656	0.008	12.5000	32.9891	10	29.9166	1
B21	5	0.585938	0.097656	0.008	12.5000	37.7747	26	29.9040	1
B22	5	0.585938	0.097656	0.008	12.5000	38.4845	29	29.9184	1

4.2.10 Summary of Denoising Quality Measures for Each Test

Tables 4.10 to 4.14 show quality measures for each test including **PSNR**, **MSE** and **TQ** and **pTQ**. Tables 4.19 and 4.20 show a summary of these quality measures for all tests. In these results the top methods are ranked in order KSS, IDF, MMW and MW, for the red and green images. For the blue images, MMW is the best, with KSS IDF and MW closely following. Specific gains in quality for PSNR, MSE and TQ for the red and green images are substantial. PSNR is increased by 27 to 38 percent for the red and green images and by 1 to 4 percent for the blue images. MSE is reduced from an initial value of $O(-4)$ to between $O(-6)$ to $O(-8)$ depending on the images. TQ shows gains between 38 to 68 percent. For the blue images corresponding gains in PSNR are around 4 percent, MSE remains in the neighborhood of the initial value at $O(-4)$ and TQ gains are from 6 to 21 percent. These results consistently show **KSSFast** yields superior denoising when compared to the other methods for the Red and Green color images, and remains competitive with the best of the methods for the blue matrix.

Table 4.11: Comparative Denoising Quality Landsat 2021-03-23 n=0.0100 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	4.6837e-04	81.4250		18.5968	33.0073	
R	IDF	4.7348e-04	81.3778	-0.00	16.6290	32.3424	-0.02
R	KSS	7.9936e-07	109.1034	0.34	52.1990	53.2298	0.61
R	MMV	3.0457e-04	83.2940	0.02	22.0432	34.7614	0.05
R	MW	4.7348e-04	81.3778	-0.00	16.6290	32.3424	-0.02
R	MS	3.0648e-03	73.2668	-0.10	22.5790	31.6301	-0.04
R	MTHs	3.0648e-03	73.2668	-0.10	22.5790	31.6301	-0.04
R	MTHh	3.0648e-03	73.2668	-0.10	22.5790	31.6301	-0.04
G	IDF	4.7853e-04	81.3317	-0.00	16.4630	32.2724	-0.01
G	KSS	4.1486e-07	111.9518	0.38	53.4181	54.5720	0.68
G	MMV	2.8927e-04	83.5177	0.03	21.9247	34.7961	0.07
G	MW	4.7853e-04	81.3317	-0.00	16.4630	32.2724	-0.01
G	MS	3.5957e-03	72.5730	-0.11	22.5503	31.3919	-0.03
G	MTHs	3.5957e-03	72.5730	-0.11	22.5503	31.3919	-0.03
G	MTHh	3.5957e-03	72.5730	-0.11	22.5503	31.3919	-0.03
B	IDF	4.8036e-04	81.3152	-0.00	9.6059	30.0041	-0.05
B	KSS	4.7116e-04	81.3991	0.00	34.8910	38.3759	0.21
B	MMV	2.2614e-04	84.5871	0.04	13.1096	32.2400	0.02
B	MW	4.8036e-04	81.3152	-0.00	9.6059	30.0041	-0.05
B	MS	1.7483e-03	75.7046	-0.07	14.4864	29.7636	-0.06
B	MTHs	1.7483e-03	75.7046	-0.07	14.4864	29.7636	-0.06
B	MTHh	1.7483e-03	75.7046	-0.07	14.4864	29.7636	-0.06

Table 4.12: Comparative Denoising Quality Landsat 2021-03-23 n=0.0044 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	2.0869e-04	84.9358		26.6488	36.8230	
R	IDF	2.1020e-04	84.9044	-0.00	22.3090	35.3805	-0.04
R	KSS	6.7408e-07	109.8437	0.29	52.5904	53.6032	0.46
R	MMV	1.5722e-04	86.1657	0.01	25.3942	36.8148	-0.00
R	MW	2.1020e-04	84.9044	-0.00	22.3090	35.3805	-0.04
R	MS	2.2525e-03	74.6041	-0.12	24.4294	32.6818	-0.11
R	MTHs	2.2525e-03	74.6041	-0.12	24.4294	32.6818	-0.11
R	MTHh	2.2525e-03	74.6041	-0.12	24.4294	32.6818	-0.11
G	IDF	2.1300e-04	84.8470	-0.00	23.1931	35.6533	-0.02
G	KSS	3.7844e-07	112.3509	0.32	53.6551	54.7820	0.51
G	MMV	1.5035e-04	86.3597	0.02	25.9074	37.0482	0.02
G	MW	2.1300e-04	84.8470	-0.00	23.1931	35.6533	-0.02
G	MS	2.7092e-03	73.8024	-0.13	24.8020	32.5403	-0.10
G	MTHs	2.7092e-03	73.8024	-0.13	24.8020	32.5403	-0.10
G	MTHh	2.7092e-03	73.8024	-0.13	24.8020	32.5403	-0.10
B	IDF	2.1384e-04	84.8300	-0.00	14.5936	32.8099	-0.08
B	KSS	2.0735e-04	84.9637	0.00	37.6429	40.4602	0.14
B	MMV	1.1565e-04	87.4992	0.03	15.7067	34.0580	-0.04
B	MW	2.1384e-04	84.8300	-0.00	14.5936	32.8099	-0.08
B	MS	1.1389e-03	77.5661	-0.09	16.2012	30.9436	-0.13
B	MTHs	1.1389e-03	77.5661	-0.09	16.2012	30.9436	-0.13
B	MTHh	1.1389e-03	77.5661	-0.09	16.2012	30.9436	-0.13

Table 4.13: Comparative Denoising Quality Peppers 2021-03-24 n=0.0100 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	5.6602e-04	80.6025		10.7682	30.1525	
R	IDF	6.0336e-04	80.3250	-0.00	10.0090	29.8104	-0.01
R	KSS	1.6982e-06	105.8309	0.31	44.9086	49.7441	0.65
R	MMV	3.0313e-04	83.3145	0.03	15.8001	32.7079	0.08
R	MW	6.0336e-04	80.3250	-0.00	10.0090	29.8104	-0.01
R	MS	3.3811e-03	72.8402	-0.10	15.3407	29.1008	-0.03
R	MTHs	3.3811e-03	72.8402	-0.10	15.3407	29.1008	-0.03
R	MTHh	3.3811e-03	72.8402	-0.10	15.3407	29.1008	-0.03
G	IDF	7.0554e-04	79.6456	-0.00	9.0281	29.2626	-0.01
G	KSS	9.3532e-07	108.4212	0.36	46.9850	51.2841	0.74
G	MMV	3.1324e-04	83.1720	0.04	15.1732	32.4540	0.10
G	MW	7.0554e-04	79.6456	-0.00	9.0281	29.2626	-0.01
G	MS	2.7506e-03	73.7365	-0.07	16.3471	29.7285	0.01
G	MTHs	2.7506e-03	73.7365	-0.07	16.3471	29.7285	0.01
G	MTHh	2.7506e-03	73.7365	-0.07	16.3471	29.7285	0.01
B	IDF	7.1834e-04	79.5675	-0.00	7.6904	28.7953	-0.02
B	KSS	8.2359e-04	78.9737	-0.01	22.3412	33.4342	0.14
B	MMV	3.1248e-04	83.1826	0.04	13.2887	31.8356	0.08
B	MW	7.1834e-04	79.5675	-0.00	7.6904	28.7953	-0.02
B	MS	2.3147e-03	74.4858	-0.07	14.3469	29.3156	-0.00
B	MTHs	2.3147e-03	74.4858	-0.07	14.3469	29.3156	-0.00
B	MTHh	2.3147e-03	74.4858	-0.07	14.3469	29.3156	-0.00

Table 4.14: Comparative Denoising Quality peppers 2021-03-23 n=0.0044 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	2.6007e-04	83.9798		16.0760	33.0185	
R	IDF	2.7234e-04	83.7797	-0.00	14.5978	32.4647	-0.02
R	KSS	1.6049e-06	106.0762	0.26	45.2108	49.9247	0.51
R	MMV	1.5764e-04	86.1542	0.03	20.0941	35.0620	0.06
R	MW	2.7234e-04	83.7797	-0.00	14.5978	32.4647	-0.02
R	MS	2.4928e-03	74.1639	-0.12	18.1472	30.4635	-0.08
R	MTHs	2.4928e-03	74.1639	-0.12	18.1472	30.4635	-0.08
R	MTHh	2.4928e-03	74.1639	-0.12	18.1472	30.4635	-0.08
G	IDF	3.1037e-04	83.2120	-0.00	13.9379	32.0596	-0.01
G	KSS	8.6996e-07	108.7358	0.31	47.1265	51.4346	0.59
G	MMV	1.5496e-04	86.2285	0.04	19.7914	34.9866	0.08
G	MW	3.1037e-04	83.2120	-0.00	13.9379	32.0596	-0.01
G	MS	1.8465e-03	75.4672	-0.09	19.7299	31.4157	-0.03
G	MTHs	1.8465e-03	75.4672	-0.09	19.7299	31.4157	-0.03
G	MTHh	1.8465e-03	75.4672	-0.09	19.7299	31.4157	-0.03
B	IDF	3.1147e-04	83.1967	-0.00	11.8284	31.3584	-0.03
B	KSS	3.4018e-04	82.8137	-0.01	27.3940	36.3687	0.12
B	MMV	1.5234e-04	86.3028	0.04	17.2379	34.1685	0.05
B	MW	3.1147e-04	83.1967	-0.00	11.8284	31.3584	-0.03
B	MS	1.4830e-03	76.4195	-0.08	17.2355	30.9066	-0.05
B	MTHs	1.4830e-03	76.4195	-0.08	17.2355	30.9066	-0.05
B	MTHh	1.4830e-03	76.4195	-0.08	17.2355	30.9066	-0.05

Table 4.15: Comparative Denoising Quality Boats 2021-03-30 n=0.0044 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	2.6436e-05	93.9088		25.0665	39.2618	
R	IDF	2.6192e-05	93.9492	0.00	24.8352	39.1989	-0.00
R	KSS	4.5699e-08	121.5317	0.29	54.2596	58.0111	0.48
R	MMV	1.5875e-05	96.1236	0.02	28.7978	41.2241	0.05
R	MW	2.6192e-05	93.9492	0.00	24.8352	39.1989	-0.00
R	MS	2.4569e-04	84.2268	-0.10	26.0981	36.4073	-0.07
R	MTHs	2.4569e-04	84.2268	-0.10	26.0981	36.4073	-0.07
R	MTHh	2.4569e-04	84.2268	-0.10	26.0981	36.4073	-0.07
G	IDF	2.6819e-05	93.8464	-0.00	25.1842	39.2801	-0.00
G	KSS	2.2639e-08	124.5823	0.33	56.8766	59.8814	0.52
G	MMV	1.6749e-05	95.8909	0.02	29.3690	41.3358	0.05
G	MW	2.6819e-05	93.8464	-0.00	25.1842	39.2801	-0.00
G	MS	2.8447e-04	83.5905	-0.11	26.7245	36.4040	-0.08
G	MTHs	2.8447e-04	83.5905	-0.11	26.7245	36.4040	-0.08
G	MTHh	2.8447e-04	83.5905	-0.11	26.7245	36.4040	-0.08
B	IDF	2.5380e-05	94.0860	0.00	25.2497	39.3808	-0.00
B	KSS	2.6674e-05	93.8699	-0.00	35.3701	42.6492	0.08
B	MMV	1.6315e-05	96.0049	0.02	29.3089	41.3536	0.05
B	MW	2.5380e-05	94.0860	0.00	25.2497	39.3808	-0.00
B	MS	2.3252e-04	84.4662	-0.10	25.8154	36.3930	-0.08
B	MTHs	2.3252e-04	84.4662	-0.10	25.8154	36.3930	-0.08
B	MTHh	2.3252e-04	84.4662	-0.10	25.8154	36.3930	-0.08

Table 4.16: Comparative Denoising Quality Boat 2021-03-30 n=0.0100 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	5.9864e-05	90.3592		19.8365	36.3646	
R	IDF	5.9474e-05	90.3875	0.00	19.6736	36.3202	-0.00
R	KSS	5.7411e-08	120.5409	0.33	54.1747	57.6561	0.59
R	MMV	3.2394e-05	93.0262	0.03	24.7849	38.8777	0.07
R	MW	5.9474e-05	90.3875	0.00	19.6736	36.3202	-0.00
R	MS	3.3518e-04	82.8780	-0.08	23.4812	35.0986	-0.03
R	MTHs	3.3518e-04	82.8780	-0.08	23.4812	35.0986	-0.03
R	MTHh	3.3518e-04	82.8780	-0.08	23.4812	35.0986	-0.03
G	IDF	5.8392e-05	90.4673	-0.00	19.9376	36.4336	-0.00
G	KSS	2.3973e-08	124.3336	0.37	57.2462	59.9214	0.64
G	MMV	3.3638e-05	92.8625	0.03	25.1721	38.9514	0.07
G	MW	5.8392e-05	90.4673	-0.00	19.9376	36.4336	-0.00
G	MS	3.7141e-04	82.4322	-0.09	24.1140	35.1604	-0.04
G	MTHs	3.7141e-04	82.4322	-0.09	24.1140	35.1604	-0.04
G	MTHh	3.7141e-04	82.4322	-0.09	24.1140	35.1604	-0.04
B	IDF	5.4123e-05	90.7970	-0.00	20.0305	36.5731	-0.00
B	KSS	5.7085e-05	90.5656	-0.00	29.5618	39.6420	0.08
B	MMV	3.2723e-05	92.9823	0.02	25.1572	38.9861	0.06
B	MW	5.4123e-05	90.7970	-0.00	20.0305	36.5731	-0.00
B	MS	3.0067e-04	83.3499	-0.08	23.4370	35.2398	-0.04
B	MTHs	3.0067e-04	83.3499	-0.08	23.4370	35.2398	-0.04
B	MTHh	3.0067e-04	83.3499	-0.08	23.4370	35.2398	-0.04

Table 4.17: Comparative Denoising Quality USM 2021-03-25 n=0.0044 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	1.1662e-04	87.4631		29.8669	38.7189	
R	IDF	1.2259e-04	87.2461	-0.00	29.2797	38.4536	-0.01
R	KSS	5.0669e-07	111.0834	0.27	51.2840	53.5813	0.38
R	MMV	9.0911e-05	88.5446	0.01	25.4825	37.6290	-0.03
R	MW	1.2259e-04	87.2461	-0.00	29.2797	38.4536	-0.01
R	MS	1.0518e-03	77.9113	-0.11	26.3295	34.3998	-0.11
R	MTHs	1.0518e-03	77.9113	-0.11	26.3295	34.3998	-0.11
R	MTHh	1.0518e-03	77.9113	-0.11	26.3295	34.3998	-0.11
G	IDF	1.3488e-04	86.8312	-0.00	29.0171	38.2300	-0.01
G	KSS	4.2061e-07	111.8920	0.29	50.4651	53.5779	0.39
G	MMV	9.3172e-05	88.4379	0.02	25.2159	37.5058	-0.02
G	MW	1.3488e-04	86.8312	-0.00	29.0171	38.2300	-0.01
G	MS	1.0346e-03	77.9829	-0.10	26.4137	34.4512	-0.10
G	MTHs	1.0346e-03	77.9829	-0.10	26.4137	34.4512	-0.10
G	MTHh	1.0346e-03	77.9829	-0.10	26.4137	34.4512	-0.10
B	IDF	1.2583e-04	87.1330	0.00	25.8049	37.2695	-0.02
B	KSS	1.2833e-04	87.0474	0.00	36.2378	40.6842	0.07
B	MMV	6.7482e-05	89.8389	0.03	21.5078	36.7445	-0.03
B	MW	1.2583e-04	87.1330	0.00	25.8049	37.2695	-0.02
B	MS	8.9532e-04	78.6110	-0.10	22.8819	33.4930	-0.12
B	MTHs	8.9532e-04	78.6110	-0.10	22.8819	33.4930	-0.12
B	MTHh	8.9532e-04	78.6110	-0.10	22.8819	33.4930	-0.12

Table 4.18: Comparative Denoising Quality USM 2021-03-25 n=0.0100 (114r)

Color	Method	mse	psnr	ppsnr	Ssim	TQ	pTQ
R	Init	2.5268e-04	84.1050		24.5578	35.8588	
R	IDF	2.5451e-04	84.0738	-0.00	24.4963	35.8282	-0.00
R	KSS	5.3937e-07	110.8119	0.32	52.1726	53.7849	0.50
R	MMV	1.8104e-04	85.5531	0.02	23.0142	35.8273	-0.00
R	MW	2.5451e-04	84.0738	-0.00	24.4963	35.8282	-0.00
R	MS	1.3927e-03	76.6923	-0.09	24.8214	33.5000	-0.07
R	MTHs	1.3927e-03	76.6923	-0.09	24.8214	33.5000	-0.07
R	MTHh	1.3927e-03	76.6923	-0.09	24.8214	33.5000	-0.07
G	IDF	2.8112e-04	83.6419	0.00	23.9982	35.5213	-0.00
G	KSS	5.2806e-07	110.9040	0.33	50.3259	53.2059	0.50
G	MMV	1.8656e-04	85.4227	0.02	22.6849	35.6756	0.00
G	MW	2.8112e-04	83.6419	0.00	23.9982	35.5213	-0.00
G	MS	1.3989e-03	76.6729	-0.08	24.9126	33.5237	-0.06
G	MTHs	1.3989e-03	76.6729	-0.08	24.9126	33.5237	-0.06
G	MTHh	1.3989e-03	76.6729	-0.08	24.9126	33.5237	-0.06
B	IDF	2.7801e-04	83.6902	0.00	20.5057	34.3847	-0.01
B	KSS	2.7706e-04	83.7050	0.00	33.6685	38.7333	0.11
B	MMV	1.4864e-04	86.4094	0.04	19.0300	34.7951	-0.00
B	MW	2.7801e-04	83.6902	0.00	20.5057	34.3847	-0.01
B	MS	1.1912e-03	77.3711	-0.07	21.2682	32.5514	-0.07
B	MTHs	1.1912e-03	77.3711	-0.07	21.2682	32.5514	-0.07
B	MTHh	1.1912e-03	77.3711	-0.07	21.2682	32.5514	-0.07

4.2.11 Tabular Results All Tests

Parameter Optimization All Tests

Table 4.19: Parameter Optimization R11 Block All Tests v114

Im	σ^2	nb	tb*5	dtb*5	epsb	kb*3	TQb	pTQ	TQ0	ub
L	10	3	0.390625	0.097656	0.008	3.1250	35.1388	25	28.2006	0
L	4.4	3	0.781250	0.195313	0.008	12.5000	37.8069	29	29.4187	1
P	10	2	1.171875	0.390625	0.008	3.1250	30.3133	10	27.4978	1
P	4.4	2	0.585938	0.195313	0.008	3.1250	33.0111	15	28.6952	1
B	10	2	0.146484	0.048828	0.008	3.1250	35.6941	18	30.3014	1
B	4.4	2	0.146484	0.048828	0.008	3.1250	37.5129	19	31.4580	1
U	10	3	0.195313	0.048828	0.008	3.1250	36.6106	27	28.9264	1
U	4.4	4	0.488281	0.097656	0.008	12.5000	39.1680	30	30.0551	1

Comparative Denoising TQ All Tests

Table 4.20: Comparative Denoising TQ All Tests (114r)

Im	$\sigma^2 * 3$	Cl	Init	IDF	KSS	MMV	MW	MS	MTHsh
L	4.4	R	36.8230	35.3805	53.6032	36.8148	35.3805	32.6818	32.6818
L	4.4	G	36.2279	35.6533	54.7820	37.0482	35.6533	32.5403	32.5403
L	4.4	B	35.4873	32.8099	40.4602	34.0580	32.8099	30.9436	30.9436
L	10	R	33.0073	32.3424	53.2298	34.7614	32.3424	31.6301	31.6301
L	10	G	32.4713	32.2724	54.5720	34.7961	32.2724	31.3919	31.3919
L	10	B	31.7355	30.0041	38.3759	32.2400	30.0041	29.7636	29.7636
P	4.4	R	33.0185	32.4647	49.9247	35.0620	32.4647	30.4635	30.4635
P	4.4	G	32.3908	32.0596	51.4346	34.9866	32.0596	31.4157	31.4157
P	4.4	B	32.3923	31.3584	36.3687	34.1685	31.3584	30.9066	30.9066
P	10	R	30.1525	29.8104	49.7441	32.7079	29.8104	29.1008	29.1008
P	10	G	29.5384	29.2626	51.2841	32.4540	29.2626	29.7285	29.7285
P	10	B	29.4334	28.7953	33.4342	31.8356	28.7953	29.3156	29.3156
B	4.4	R	39.2618	39.1989	58.0111	41.2241	39.1989	36.4073	36.4073
B	4.4	G	39.3563	39.2801	59.8814	41.3358	39.2801	36.4040	36.4040
B	4.4	B	39.4706	39.3808	42.6492	41.3536	39.3808	36.3930	36.3930
B	10	R	36.3646	36.3202	57.6561	38.8777	36.3202	35.0986	35.0986
B	10	G	36.4827	36.4336	59.9214	38.9514	36.4336	35.1604	35.1604
B	10	B	36.6594	36.5731	39.6420	38.9861	36.5731	35.2398	35.2398
U	4.4	R	38.7189	38.4536	53.5813	37.6290	38.4536	34.3998	34.3998
U	4.4	G	38.4327	38.2300	53.5779	37.5058	38.2300	34.4512	34.4512
U	4.4	B	37.9983	37.2695	40.6842	36.7445	37.2695	33.4930	33.4930
U	10	R	35.8588	35.8282	53.7849	35.8273	35.8282	33.5000	33.5000
U	10	G	35.5419	35.5213	53.2059	35.6756	35.5213	33.5237	33.5237
U	10	B	34.8780	34.3847	38.7333	34.7951	34.3847	32.5514	32.5514

Comparative Denoising PSNR All Tests

Table 4.21: Comparative Denoising Quality PSNR All Tests (114r)

Im	$\sigma^2 * 3$	Cl	Init	IDF	KSS	MMV	MW	MS	MTHsh
L	4.4	R	84.9358	84.9044	109.8437	86.1657	84.9044	74.6041	74.6041
L	4.4	G	84.9200	84.8470	112.3509	86.3597	84.8470	73.8024	73.8024
L	4.4	B	84.9300	84.8300	84.9637	87.4992	84.8300	77.5661	77.5661
L	10	R	81.4250	81.3778	109.1034	83.2940	81.3778	73.2668	73.2668
L	10	G	81.4011	81.3317	111.9518	83.5177	81.3317	72.5730	72.5730
L	10	B	81.3896	81.3152	81.3991	84.5871	81.3152	75.7046	75.7046
P	4.4	R	83.9798	83.7797	106.0762	86.1542	83.7797	74.1639	74.1639
P	4.4	G	83.2229	83.2120	108.7358	86.2285	83.2120	75.4672	75.4672
P	4.4	B	83.3027	83.1967	82.8137	86.3028	83.1967	76.4195	76.4195
P	10	R	80.6025	80.3250	105.8309	83.3145	80.3250	72.8402	72.8402
P	10	G	79.6696	79.6456	108.4212	83.1720	79.6456	73.7365	73.7365
P	10	B	79.6882	79.5675	78.9737	83.1826	79.5675	74.4858	74.4858
B	4.4	R	93.9088	93.9492	121.5317	96.1236	93.9492	84.2268	84.2268
B	4.4	G	93.9287	93.8464	124.5823	95.8909	93.8464	83.5905	83.5905
B	4.4	B	94.0754	94.0860	93.8699	96.0049	94.0860	84.4662	84.4662
B	10	R	90.3592	90.3875	120.5409	93.0262	90.3875	82.8780	82.8780
B	10	G	90.5373	90.4673	124.3336	92.8625	90.4673	82.4322	82.4322
B	10	B	90.9112	90.7970	90.5656	92.9823	90.7970	83.3499	83.3499
U	4.4	R	87.4631	87.2461	111.0834	88.5446	87.2461	77.9113	77.9113
U	4.4	G	86.9473	86.8312	111.8920	88.4379	86.8312	77.9829	77.9829
U	4.4	B	86.9022	87.1330	87.0474	89.8389	87.1330	78.6110	78.6110
U	10	R	84.1050	84.0738	110.8119	85.5531	84.0738	76.6923	76.6923
U	10	G	83.5741	83.6419	110.9040	85.4227	83.6419	76.6729	76.6729
U	10	B	83.4144	83.6902	83.7050	86.4094	83.6902	77.3711	77.3711

4.2.12 Parameter Optimization and Denoising Time All Tests

Table 4.21 shows parameter optimization and denoising time for combined for the **KSS** method, and denoising time for six comparative methods. To interpret the times fairly recall that the **KSSFast** includes parameter optimization for a single block, and denoising of each block utilizing the **KSSFast** solver for repeated iterations. This explains why the **KSS** method times are significantly higher than the comparative methods which are implemented with a single **MATLAB** command to denoise the whole color images. Parameter optimization and denoising time for the first red block takes about 120 seconds and denoising each other block takes about 30 seconds for a total processing time of about 9 minutes per image, the overall runtime of the code is about 20 minutes per which includes slowdown for printing to monitor function for generating tables and figures for of results. These per image times can be reduced through improvements in code organization. As the proposed parameter optimization and denoising method is compared all other denoising methods each take less than 1 second to run on each whole color image. The KSS time measures are dominated by the parameter optimization time for the first block and thus **KSSFast** is slow in comparison to the other denoising methods. But since this time includes parameter optimization and denoising for each block it can be considered a worthwhile investment through tuning parameters used in **KSSFast** results in superior denoising as has been shown in all tests. In conclusion the algorithm efficiency of the code using multiple calls to **KSSFast** in this research cannot fairly be compared against a single call of **MATLAB** commands. Each **MATLAB** command has been optimized for efficiency and minimal runtime, whereas this implementation of **KSSFast** in conjunction with parameter optimization is still in a development.

Table 4.22: Comparative Denoising Time All Tests (s)

Image	Color	$\sigma^2 * 3$	MTHs	MTHh	MMV	MW	MS	IDF	KSS
L	R	10	0.0013	0.0007	0.0032	0.1662	0.0008	0.1393	300
L	G	10	0.0012	0.0006	0.0029	0.1634	0.0009	0.1420	120
L	B	10	0.0012	0.0006	0.0041	0.1742	0.0011	0.0719	120
L	R	4.4	0.0014	0.0007	0.0039	0.2022	0.0012	0.1819	300
L	G	4.4	0.0017	0.0009	0.0045	0.2175	0.0011	0.1846	120
L	B	4.4	0.0018	0.0008	0.0043	0.2095	0.0012	0.1051	120
P	R	10	0.0016	0.0008	0.0048	0.1963	0.0009	0.1185	300
P	G	10	0.0010	0.0005	0.0025	0.0836	0.0008	0.0709	120
P	B	10	0.0010	0.0004	0.0027	0.1039	0.0008	0.0457	120
P	R	4.4	0.0010	0.0005	0.0029	0.0934	0.0007	0.0964	300
P	G	4.4	0.0010	0.0004	0.0027	0.0936	0.0008	0.0894	120
P	B	4.4	0.0016	0.0010	0.0058	0.1664	0.0009	0.0888	120
U	R	10	0.0017	0.0009	0.0053	0.5449	0.0014	0.2518	300
U	G	10	0.0017	0.0007	0.0054	0.5288	0.0014	0.2400	120
U	B	10	0.0016	0.0008	0.0050	0.5216	0.0015	0.2567	120
U	R	4.4	0.0015	0.0007	0.0051	0.5388	0.0016	0.2365	300
U	G	4.4	0.0018	0.0009	0.0048	0.5271	0.0017	0.2379	120
U	B	4.4	0.0030	0.0022	0.0074	0.6720	0.0018	0.2665	120

Chapter 5
Conclusions and Future Work

5.1 Conclusions of Research

Testing of the four images at two noise levels has provided a challenging denoising test for this research. Parameter optimization customized the denoising of each image in a localized way to achieve superior denoising through application of the **KSSFast** solver to each subblock independently. Figures and numerical results demonstrate that **KSSFast** is an effective denoising tool that meets or exceeds the perfomance of the other denoising methods it is compared with herein. Images are denoised very well for the red and green images, and as good as the competition for the blue images. The recombined RGB images are of high quality. This research has shown the parameter optimization combined with localized denoising of each color image subblock with the **KSSFast** solver is an important contribution to image denoising research.

5.2 Future Work

Future work in four areas to pursue observations made during the tests for this research. The perspective gained from denoising single color subblocks identified issues with blue blur in images having low pixel intensity or inconsistent color balance in an image which if addressed could lead to blue blur reduction. Observations of the Fourier transform coefficients during denoising showed a correlation between noise removal and the decay of Fourier coefficients to achieve superior sharpening with a variable frequency based regularization parameter based upon frequency and also predict of the ideal quality state which would reduce the number **KSSFast** iterations to improve algorithm efficiency. Finally implementation of parallel processing would reduce denoising time if all subblocks were processed simultaneously.

- Reducing Blue Blur With Color Rebalancing
- A Variable Regularization Parameter based on Fourier Transform CoefficientDecay
- Optimum Quality State Predicted From Fourier Coefficient Decay Rates
- Parallel Processing to Denoise all Sublocks Simultaneously

BIBLIOGRAPHY

[1] Carrington, D. : How Many Photos Will Be Taken in 2020?,*Life In Focus,MYLIO.COM* (2020).

[2] Golub, G. H. and Welsch, J.: Calculation of Gauss quadrature rules, *Math. Comp.* **23** (1969) 221âĂŞ230.

[3] Golub, G. H. and Meurant, C.: Matrices, Moments and Quadrature. *Proceedings of the 15th Dundee Conference*, June-July 1993, D. F. Griffiths and G. A. Watson (eds.), Longman Scientific & Technical (1994)

[4] Hochbruck, M. and Lubich, C.: On Krylov subspace approximation to the matrix exponential operator. *SIAM J. Numer. Anal.* **28** (2008) 114-135.

[5] Lambers, J. V.: Enhancement of Krylov Subspace Spectral Methods by Block Lanczos Iteration. *Electronic Transactions on Numerical Analysis* **31** (2008) 86-109.

[6] Palchak, E. M., Cibotarica, A. and Lambers, J. V.: Solution of Time-Dependent PDE Through Rapid Estimation of Block Gaussian Quadrature Nodes. *Linear Algebra and its Applications* **468** (2015) 233-259.

[7] 7. Lambers, J. V.: Spectral Methods for Time-dependent Variable-coefficient PDE Based on Block Gaussian Quadrature. *Proceedings of the World Congress on Engineering* London (2010)

[8] 8. Lambers, J.V.: Krylov Subspace Methods for Variable-Coefficient Initial-Boundary Value Problems. Ph.D. Thesis. Stanford University, SCCM Program, (2003)

[9] 9. Lambers, J. V.: Krylov subspace spectral methods for variable-coefficient initial-boundary value problems. *Electron Trans. Numer. Anal.* **20** (2005) 212âĂŞ234.

[10] Lambers, J. V. and Sumner, A. C.: *Explorations in Numerical Analysis*, New Jersey, World Scientific (2018) 289-326.

[11] LeVeque, Randall J.: *Finite difference methods for ordinary and partial differential equations: steady-State and time-Dependent problems*, Society for Industrial and Applied Mathematics (2007)

[12] Cibotarica, A., Lambers, J. V. and Palchak, E. M.: Solution of Nonlinear Time-Dependent PDE Through Componentwise Approximation of Matrix Functions. *Journal of Computational Physics* **321** (2016) 1120-1143.

[13] Pinchover, Y. and Rubinstein, J.: *An Introduction to Partial Differential Equations*, Cambridge University Press (2005)

[14] (2020) [Picture of University of Southern Mississippi Administration Building [Photograph] USM https://www.usm.edu/USM2020.jpg.

[15] Golub, G. H., Underwood, R.: The Block Lanczos method for computing eigenvalues. *Mathematical Software III*, J. Rice Ed., (1977) 361-377.

[16] Liesen, J. , Strakos, Z.,: Krylov Subspace Methods Priciples and Analysis. *Numerical Mathematics and Scientific Computation* (2013) (19-23)

[17] Fan, L., Zhang, F., Fan H., and Zhang, C.: Brief review of image denoising techniques. *Visual Computing for Industry, Biomedicine, and Art* 2:7 (2019).

[18] Buades,A.,Coll, B., and Morel,J.M.: A Review of Image Denoising Algorithms, with a New One. *SIAM Multiscale Model Simulation* **4** (2) 490-530.

[19] Guidotti, P., Lambers,J.V.,: Two New Nonlinear Nonlocal Diffusions for Noise Reduction. J math Imageing Vis **33** 25-37 (2009)

[20] Perona, P., Malik, J.: Scale-space and edge detection using anisotropic diffusion.

[21] Kichenassamy, S.: The Perona-Malk Paradox. SIAM J. Appl. Math. **57** 1328-1342 (1997)

[22] You,Y.L., Kaveh, M.: Fourth-order partial differential equations for noise removal. *IEEE Transactions Image Processing* **9**(10),1723-1730 (2000) *IEEE Trans. Pattern Anal. Mach . Intell.* **12** 161-192 (1990)

[23] R2020a MATLAB Image Processing Toolbox Test Images, Peppers. *The MathWorks, Inc.* (2020)

[24] Lambers, J.V.,: Enhancement of Krylov Subspace Spectral Methods By Block Lanczos Iteration. *Electronic Transactions on Numerical Analysis*. (2008) (31)

[25] R2020a MATLAB Image Processing Toolbox Documentation *The MathWorks, Inc.* (2020)

[26] Wang,Z., Bovik, A.C., Sheikh,H.R. and Simoncelli,E.P.: Image Quality Assessment: From Error Visibility to Structural Similarity *IEEE Transactions on Image Processing* **13** (4) 600- 612(2004)

[27] Guidotti, P., Longo, K.: Two Enhanced Fourth Order Diffusion Models for Image Denoising. *Journal of Mathematical Imagining and Vision* (?)

[28] Guidotti, P., Kim, Y., Lambers, J.V.: Image restoration with a new class of forward-backward-forward diffusion equations of Perona-Malik type with Application to Satellite Image Enhancement. SIAM Journal on Imaging Sciences 6(3) (2013) 1416-1444

[29] Duong, L.: Efficient Denoising and Sharpening of Color Images through numerical solution of nonlinear diffusion equation. *The University of Southern Mississippi The Aquila Digital Community Undergraduate Honors Thesis*

[30] Guidotti, P. : A new Nonlocal Nonlinear Diffusion of Image Processing. *Journal of Differential Equations* **246**(12) 4731-4742.

[31] Guidotti, P. : A New well-posed Nonlinear Nonlocal Diffusion. *Journal of Mathematical Imagine and Vision* **72** 4625-4637

[32] Weickert, J.: Anisotropic Diffusion in Image Processing. *ECMI Series Teubner Verlag, Suttgart (1998)*

[33] Mathieu, B., Melchior, P., Outstaloup, A.c, Ceyral, C.: Fractional Differentiation for Edge Detection. Signal Processing 83 2421-2432 (2003)

[34] Venkatanath, N., Praneeth, D., Chandrasekhar, M., Bh, Channappayya, S., and Medasani, S.,: Blind image quality evaluation using perception based features. *2015 Twenty First National Conference on Communications IEEE* (2015) 1-6

[35] Witkin, A. P. : Scale-space filtering *Proc. 8th Int. Joint Conf. Art. Intell.*1019–1022, (1983)

[36] Babaud, A., Witkin, M., and Duda, R.: Uniqueness of the gaussian kernel for scal-space filtering. Trans. Pattern Anal. Machine Intell., -8, (1986)

[37] Irum,I.,Shahid, M.A., Sharif, M. and Raza, M.: A Review of Image Denoising Methods. *Journal of Engineering Sciencs and Technology Review* (2015)

[38] Yang,Q.,Chen,D., Zhao,T.,Chen,Y.,: Fractional Calculaus In Image Processing: A Review *arXiv* (2016)

[39] da Silva,E., Medonca,G.: The Electrical Engineering Handbook-Digital Image Processing *Elsevier Science Direct* (2005) 891-893.

[40] Lillesand T., Kiefer, R.,: Remote Sensing and Image Interpretation (1994) 536.

[41] 41. Irum,I., Shahid,M.,Sharif,M., Raza,M.: A Review of Image Denoising Methods *Journal of Engineering Science and Technology Review* (2015)

[42] Koenderink,J., : The Structure of Images *Biol. Cybern.* (**50**) (1984) 3363-370.
remove any ref to 43 except the image peppers

[43] The Mathworks website., : *Image Processing Toolbox Documentation (R2020A)* (2020). http://www.mathwork.com/access/helpdesk/hel/toolbox/ images.

[44] Landsat8., (2018) [Picture of Theewaterskloof reservoir in South Africa] [Photograph] U.S. Geological Survey

`https://www.usgs.gov/media/images/theewaterskloof-reservoir-south-africa/IOW_CLU_CapeTown.jpg`.

[45] Bingham, B.,: Scalable Time-Stepping for Navier-Stokes Through High Frequency Analysis of Block Arnoldi Iteration. *University of Southern Mississippi,The Aquila Digital Community, Doctoral Disseration* (2019)

[46] Lambers, J.V.,: Krylov Subspace Spectral Methods for Variable-coefficient initial-boundary value problems.: *Electronic Transactions on Numerical Analysis* **20** (2005) 212-234.

[47] Guidotti, P., : A Family of Nonlinear Diffusions Connecting Perona-Malik to Standard Diffusion. *Discrete and Continuous Dynamical Systems* (S,3) (2012)

[48] Zhu, X., Milanfar, P., : Automatic Parameter Selection for Denoising Algorithms Using a No-Reference Measure of Image Content.: *IEEE Transactions on Image Processing* (19) (2010)

[49] Jain, A., : Fundamentals of Digital Image Processing.: *Prentice Hall* (1989) 150-153

[50] Hhan, X., Sun,J., D., Guo,Z., :Multiplicative Noise Removal Based on the Smooth Diffusion Equation.: *Journal of Mathematical Imaging and Vision* (2019)

Chapter 6

Appendix Derivation of Lanczos Recursion Coefficients

6.1 The Lanczos Algorithm

The Lanczos algorithm applied to A with initial vectors u and v computes recursion coefficients α_k and β_k from an initial residual vector r_0 set to the basis function, a defined inner product, norm and Linear operator L for the PDE being solved. The algorithm psuedo code is shown below.

$r_0 = u$

$x_0 = 0$

for $k = 1, n$

$\beta_{k-1} = ||r_{k-1}||$

$x_k = rk - 1/\beta_{k-1}$

$v_k = Lx_k$

$\alpha_k = \langle x_k, v_k \rangle$

$r_k = v_k - \alpha_k x_k - \beta_{k-1} x_{k-1}$

end

6.2 Summary of Lanczos Derived Coefficients

This section summarizes the derivation of Lanczos recursion coefficients found in the sections following. For our PDE problem we have a solution of the form $u(x,y,t) = \Sigma_{\omega_1,\omega_2} u(t)_{\omega_1,\omega_2} \cos(w_1 x)\cos(w_2 y)$, and a linear operator $L = g(x,y)\Delta$. The Lanczos iteration begins with the initial residual vector initial vector r_0 set to our basis function with initial condition x_0 shown as follows:

for $k = 0$:

$r_0 = \cos(w_1 x)\cos(w_2 y)$
$x_0 = 0$
$\beta_0 = \frac{\pi}{4\sqrt{w_1 w_2}}$

for $k = 1$:

$x_1 = \frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y)$
$v_1 = g(x,y)\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y)$
$\alpha_1 = -4\sqrt{w_1 w_2}(w_1^2+w_2^2)\bar{g}$
$r_1 \frac{-4(w_1^2+w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y)[g(x,y)-B]$,
where the constant $B = +4\sqrt{w_1 w_2}\bar{g}$
$\beta_1 = B\sqrt{D(\bar{g}^2+\bar{g}+1)}$.
where $D = 4(w_1 w_2)(w_1^2+w_2^2)^2$

for $k = 2$:

$x_2 = E\cos(w_1 x)\cos(w_2 y)[g(x,y)-B]$,
where $E = \frac{-4(w_1^2+w_2^2)}{\pi[DB^2[\bar{g}^2+\bar{g}+1]}$
$v_2 = 2E[F + g_{xx(x,y)} + g_{yy}(x,y)]g(x,y)\cos(w_x x)\cos w_2 y)$,
where $F = (\frac{B}{2}-1)(w_1^2+w_2^2)$
$\alpha_2 = \frac{H^2}{2}[(\bar{g}^2 - B\bar{g})(F+\bar{g}_{xx}+\bar{g}_{yy})]$,
where $H = \frac{E}{\pi} = \frac{-4(w_1^2+w_2^2)}{[DB^2[\bar{g}^2+\bar{g}+1]}$.

6.3 First $k = 1$ Lanczos Iteration derives β_0, x_1, v_1, α_1, r_1 and β_1

6.3.1 Deriving β_0

$$\beta_0 = ||r_o|| \tag{6.1}$$

$$= \sqrt{\langle r_0, r_0 \rangle} \tag{6.2}$$

$$= \sqrt{\langle \cos(w_1 x)\cos(w_2 y), \cos(w_1 x)\cos(w_2 y) \rangle} \tag{6.3}$$

$$= \sqrt{\int_0^\pi [\int_0^\pi \cos^2(w_2 y)\cos^2(w_1 x) dx] dy}, \tag{6.4}$$

by using the half-angle identity $\cos^2(w_1 x) = \frac{1}{2}(\cos(2w_1 x) + 1)$ we have

$$= \sqrt{\int_0^\pi \frac{1}{2}(\cos(2w_2 y) + 1)[\int_0^\pi \frac{1}{2}(\cos(2w_1 x) + 1) dx] dy} \tag{6.5}$$

$$= \sqrt{\int_0^\pi \frac{1}{2}(\cos(2w_2 y) + 1)\frac{1}{2}[\int_0^\pi \frac{1}{2w_1} 2w_1 \cos(2w_1 x) + 1 dx] dy} \tag{6.6}$$

$$= \sqrt{\int_0^\pi \frac{1}{2}(\cos(2w_2 y) + 1)\frac{1}{2}[\frac{1}{2w_1}\sin(2w_1 x) + x]|_0^\pi dy} \tag{6.7}$$

by periodicity because each w_1 in an integer $\sin(2w_1\pi) = \sin(0) = 0$, so we have

$$= \sqrt{\int_0^\pi \frac{1}{2}(\cos(2w_2 y) + 1)\frac{1}{2}(\frac{\pi}{2w_1}) dy} \tag{6.8}$$

$$= \sqrt{\int_0^\pi \frac{1}{2}(\cos(2w_2 y) + 1)(\frac{\pi}{4w_1}) dy} \tag{6.9}$$

integrating the y integral similarly we get

$$= \sqrt{\frac{\pi}{4w_2}\frac{\pi}{4w_1}} \tag{6.10}$$

$$= \sqrt{\frac{\pi^2}{16 w_2 w_2 y}} \tag{6.11}$$

and finally

$$\beta_0 = \frac{\pi}{4\sqrt{w_1 w_2}} \tag{6.12}$$

6.3.2 Deriving x_1

$$x_1 = \frac{r_0}{\beta_0} \tag{6.13}$$

recall $r_0 = \cos(w_1 x)\cos(w_2 y)$ and $\beta_0 = \frac{\pi}{4\sqrt{w_1 w_2}}$

$$x_1 = \frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y) \tag{6.14}$$

6.3.3 Deriving v_1

$$v_1 = Lx_1 \tag{6.15}$$

where $L = g(x,y)\Delta$

$$= g(x,y)\Delta x_1 \tag{6.16}$$

$$= g(x,y)\frac{d}{dx}[\frac{d}{dx}x_1] + \frac{d}{dy}[\frac{d}{dy}x_1] \tag{6.17}$$

$$= g(x,y)\frac{d}{dx}[\frac{d}{dx}\frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y)] + \frac{d}{dy}[\frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y)] \tag{6.18}$$

$$= g(x,y)\frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_2 y)\frac{d}{dx}[\frac{d}{dx}\cos(w_1 x)] + \frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\frac{d}{dy}[\frac{d}{dy}\cos(w_2 y)] \tag{6.19}$$

$$= g(x,y)\frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_2 y)(-w_1^2)\cos(w_1 x)] + \frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)(-w_2^2)\cos(w_2 y)] \tag{6.20}$$

$$v_1 = g(x,y)\frac{-4\sqrt{w_1 w_2}(w_1^2 + w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y) \tag{6.21}$$

6.3.4 Deriving α_1

$$\alpha_1 = < x_1, v_1 > \qquad (6.22)$$

Recall that $x_1 = \frac{4w_1w_2}{\pi}\cos(w_1x)\cos(w_2y)$ and $v_1 = g(x,y)\frac{-4\sqrt{w_1w_2}}{\pi}(w_1^2+w_2^2)\cos(w_1x)\cos(w_2y)$.

$$\alpha_1 = \langle \frac{4w_1w_2}{\pi}\cos(w_1x)\cos(w_2y), g(x,y)\frac{-4\sqrt{w_1w_2}}{\pi}(w_1^2+w_2^2)\pi\cos(w_1x)\cos(w_2y)\rangle \qquad (6.23)$$

$$= \int_0^\pi\int_0^\pi [\frac{4w_1w_2}{\pi}\cos(w_1x)\cos(w_2y)][g(x,y)\frac{-4\sqrt{w_1w_2}}{\pi}(w_1^2+w_2^2)\pi\cos(w_1x)\cos(w_2y)]dxdy \qquad (6.24)$$

$$= \frac{-16\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi (g(x,y)\cos^2(w_1x)\cos^2(w_2y)g(x,y)dxdy \qquad (6.25)$$

$$= \frac{-16\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi (\cos^2(w_1x)\cos^2(w_2y)g(x,y)dxdy \qquad (6.26)$$

Using the half angle cosine identities: $\cos^2(w_1x) = \frac{1+\cos(2w_1x)}{2}$ and $\cos^2(w_2y) = \frac{1+\cos(2w_2y)}{2}$

$$= \frac{-16\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi (\frac{1+\cos(2w_1x)}{2})(\frac{(1+\cos(2w_2y)}{2})g(x,y)dxdy \qquad (6.27)$$

$$= \frac{-4\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi [1+\cos(2w_1x)+\cos(2w_2y)+\cos(2w_1x)\cos(2w_2y)]g(x,y)dxdy. \qquad (6.28)$$

Separating into 4 integrals

$I_1 = \frac{-4\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi g(x,y)dxdy$
$I_2 = \frac{-4\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi \cos(2w_1x)g(x,y)dxdy$
$I_3 = \frac{-4\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi \cos(2w_2y)g(x,y)dxdy$
$I_4 = \frac{-4\sqrt{w_1w_2}(w_1^2+w_2^2)}{\pi^2}\int_0^\pi\int_0^\pi \cos(2w_1x)\cos(2w_2y)g(x,y)dxdy.$
We define the average value of g on the domain as $\overline{g} = \frac{1}{\pi^2}\int_0^\pi\int_0^\pi g(x,y)dxdy$

$I_1 = -4\sqrt{w_1w_2}(w_1^2+w_2^2)\overline{g}$
$I_2 = -4\sqrt{w_1w_2}(w_1^2+w_2^2)\frac{1}{\pi^2}\int_0^\pi\int_0^\pi \cos(2w_1x)g(x,y)dxdy$

$I_3 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\frac{1}{\pi^2}\int_0^\pi \int_0^\pi \cos(2w_2 y) g(x,y)dxdy$
$I_4 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\frac{1}{\pi^2}\int_0^\pi \int_0^\pi \cos(2w_1 x)\cos(2w_2 y) g(x,y)dxdy.$

Using the defined Fourier transform coefficient $\hat{g}(w_1, w_2)] = \frac{1}{\pi^2}\int_0^\pi \int_0^\pi \cos(2w_1 x)\cos(2w_2 y) g(x,y)dxdy$ allows simplification to

$I_1 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\overline{g}$
$I_2 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\int_0^\pi \int_0^\pi \cos(2w_1 x) g(x,y)dxdy$
$I_3 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\int_0^\pi \int_0^\pi \cos(2w_2 y) g(x,y)dxdy$
$I_4 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\hat{g}(w_1, w_2).$
The middle integrals are the Fourier transform coefficients when $x = 0$ or $y = 0$ such that
$\hat{g}(w_1, 0) = \frac{1}{\pi^2}\int_0^\pi \int_0^\pi \cos(2w_1 x)\cos 0) g(x,y)dxdy$
and $\hat{g}(0, w_2) = \frac{1}{\pi^2}\int_0^\pi \int_0^\pi \cos 0)\cos(2w_2 y) g(x,y)dxdy$
and the integrals simplify to

$I_1 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\overline{g}$
$I_2 - 4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\hat{g}(w_1, 0)dxdy$
$I_3 - 4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\hat{g}(0, w_2,)dxdy$
$I_4 - 4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\hat{g}(w_1, w_2)].$
Recombining the integrals yields

$$\alpha_1 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)[\overline{g} + \hat{g}(w_1, 0) + \hat{g}(0, w_2) + \hat{g}(w_1, w_2)]. \tag{6.29}$$

All Fourier transform coefficients can be considered negligible as w_1 and w_2 approach infinity so the last three integrals in α_1 can be ignored and we finish with

$$\alpha_1 = -4\sqrt{w_1 w_2}(w_1^2 + w_2^2)\overline{g} \tag{6.30}$$

6.3.5 Deriving r_1

$$r_1 = v_1 - \alpha_1 x_1 - \beta_0 x_0 \tag{6.31}$$

Recall that $x_0 = 0$,
$\beta_0 = \frac{\pi}{4\sqrt{w_1 w_2}}$,
$x_1 = \frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y)$,
$v_1 = g(x,y)\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y)$
and $\alpha_1 = -4\sqrt{w_1 w_2}(w_1^2+w_2^2)\overline{g}$
after substitution into the equation gives

$$r_1 = [g(x,y)\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y)] - [-4\sqrt{w_1 w_2}(w_1^2+w_2^2)\overline{g}][\frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y)]. \tag{6.32}$$

After combining constants yields

$$= \frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y)[g(x,y) - 4\sqrt{w_1 w_2}\bar{g}]. \tag{6.33}$$

We define a constant $B = 4\sqrt{w_1 w_2}\bar{g}$, so that we have

$$r_1 = \frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}\cos(w_1 x)\cos(w_2 y)[g(x,y) - B]. \tag{6.34}$$

6.3.6 Deriving β_1

$$\beta_1 = ||r_1|| = \sqrt{\langle r_1, r_1\rangle} \tag{6.35}$$

Using r_1 and the defined inner product $\langle u,u\rangle = \int_0^\pi \int_0^\pi u^2 dydx$,
we will derive β_1^2 and then apply the square root

$$\beta_1^2 = ||r_1|| = \langle r_1, r_1\rangle \tag{6.36}$$

$$= \int_0^\pi \int_0^\pi [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)]}{\pi}\cos(w_1 x)\cos(w_2 y)[g(x,y) - B])^2 dydx \tag{6.37}$$

Squaring terms and collecting constants outside the integral gives

$$= [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \cos^2(w_1 x)\cos^2(w_2 y)[g(x,y) - B]^2) dydx. \tag{6.38}$$

Expanding the product in last term,

$$= [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \cos^2(w_1 x)\cos^2(w_2 y)[g(x,y)^2 - 2g(x,y)B + B]^2) dydx. \tag{6.39}$$

Separating into three integrals,

$I_1 = [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \cos^2(w_1 x)\cos^2(w_2 y)[g(x,y)^2]) dydx$
$I_2 = [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \cos^2(w_1 x)\cos^2(w_2 y)[-2Bg(x,y)]) dydx$

$I_3 = [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \cos^2(w_1 x)\cos^2(w_2 y)[B^2])dydx$

On each integral we will use the half angle formulas
$\cos^2(w_1 x) = \frac{1+\cos(2w_1 x)}{2}$ and $\cos^2(w_2 y) = \frac{1+\cos(2w_2 y)}{2}$ to get

$I_1 = [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \frac{1}{2}[g(x,y)^2](\cos 2(w_1 x)+1)\frac{1}{2}(\cos 2(w_2 y)+1))dydx$
$I_2 = [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi [-2Bg(x,y)])\frac{1}{2}(\cos 2(w_1 x)+1)\frac{1}{2}(\cos 2(w_2 y)+1)dydx$
$I_3 = [\frac{-4\sqrt{w_1 w_2}(w_1^2+w_2^2)}{\pi}]^2 \int_0^\pi \int_0^\pi \frac{1}{2}[B^2](\cos 2(w_1 x)+1)\frac{1}{2}(\cos 2(w_2 y)dydx.$

Combining constants we get

$I_1 = [\frac{4(w_1 w_2)(w_1^2+w_2^2)^2}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2](\cos 2(w_1 x)+1)(\cos 2(w_2 y)+1))dydx$
$I_2 = [\frac{-8(w_1 w_2)(w_1^2+w_2^2)^2}{\pi^2}] \int_0^\pi \int_0^\pi [Bg(x,y)])(\cos 2(w_1 x)+1)(\cos 2(w_2 y)+1)dydx$
$I_3 = [\frac{4(w_1 w_2)(w_1^2+w_2^2)^2}{\pi^2}] \int_0^\pi \int_0^\pi [B^2](\cos 2(w_1 x)+1)(\cos 2(w_2 y)dydx.$

Multiplying factors, moving B^2 and B, joining the constants in the second and third integrals and defining another constant $D = 4(w_1 w_2)(w_1^2+w_2^2)^2$
yeilds
$I_1 = [\frac{D}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2][\cos 2(w_1 x)\cos 2(w_2 y)+\cos 2(w_2 y)+\cos 2(w_1 x)+1)]dydx$
$I_2 = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)][\cos 2(w_1 x)\cos 2(w_2 y)+\cos 2(w_2 y)+\cos 2(w_1 x)+1)]dydx$
$I_3 = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [\cos 2(w_1 x)\cos 2(w_2 y)+\cos 2(w_2 y)+\cos 2(w_1 x)+1)]dydx$

Separating each of the three integrals above into four integrals identifying them by a revised index we get
$I_1 = [\frac{D}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2][\cos 2(w_1 x)\cos 2(w_2 y)]dydx$
$I_2 = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2][\cos 2(w_1 x)]dydx$
$I_3 = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2][\cos 2(w_2 y)]dydx$
$I_4 = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2][1]dydx$
$I_5 = [\frac{D}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)][\cos 2(w_1 x)\cos 2(w_{12} y)]dydx$
$I_6 = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)][\cos 2(w_2 y)]dydx$
$I_7 = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)][\cos 2(w_1 x)]dydx$
$I_8 = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)][1]dydx$
$I_9 = [\frac{D}{\pi^2}] \int_0^\pi \int_0^\pi [1][\cos 2(w_1 x)\cos 2(w_2 y)]dydx$
$I_{10} = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [1][\cos 2(w_1 x)]dydx$
$I_{11} = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [1][\cos 2(w_2 y)]dydx$
$I_{12} = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [1]dydx$

Using the definition of the Fourier cosine coefficient we can resolve integrals $1-3$ and $5-7$ as follows,

$I_1 = D\hat{g}^2(w_1, w_2)$
$I_2 = -2DB\hat{g}^2(w_1, 0)$
$I_3 = -2DB\hat{g}^2(0, w_2)$
$I_5 = D\hat{g}(w_1, w_2)$

$I_6 = -2DB\hat{g}(w_1, 0)$
$I_7 = -2DB\hat{g}(0, w_2)$

Integrals 4 and 8 integrals are the average values of g and g^2 on the domain

$I_4 = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)^2][1]dydx = DB^2\bar{g}^2(x,y)$
$I_8 = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [g(x,y)][1]dydx = DB^2\bar{g}(x,y)$
the twelfth integral evaluates to a constant
$I_{12} = [\frac{DB^2}{\pi^2}] \int_0^\pi \int_0^\pi [1]dydx = [\frac{DB^2}{\pi^2}][\pi^2 = DB^2$

Evaluating the inner integrals for integrals 9, 19 and 11 we have
$I_9 = [\frac{D}{\pi^2}] \int_0^\pi \int_0^\pi [1][\frac{1}{2w_1} \cos 2(w_1 x) \cos 2(w_2 y))]dy\ I_{10} = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [1][\cos 2(w_1 x))]dydx$

$I_{11} = [\frac{-2DB}{\pi^2}] \int_0^\pi \int_0^\pi [1][\cos 2(w_2 y))]dydx.$
Evaluating the outer integral for integrals 9, 10 and 11 by periodicity to have zero values, because as the inner integral is evaluated at π and 0 $\sin = 0$ we get
$I_9 = [\frac{D}{\pi^2}] \int_0^\pi [0] \cos 2(w_2 y))]dydx = 0$

$I_{10} = [\frac{-2DB}{\pi^2}] \int_0^\pi \frac{1}{2w_1} \sin 2(w_1 x)|_0^\pi)]dy = 0$

$I_{11} = [\frac{-2DB}{\pi^2}] \int_0^\pi \frac{1}{2w_2} \sin 2(w_2 y)|_0^\pi)]dy = 0.$

Combining the twelve integral results we get
equation$\beta_1^2 = D[\hat{g}^2(w_1, w_2)^2 + \hat{g}^2(w_1, w_2)] - 2DB[[\hat{g}^2(w_1, 0) + \hat{g}^2(0, w_2) + \hat{g}(w_1, 0) + \hat{g}(0, w_2)] + DB^2[\bar{g}^2 + \bar{g} + 1]$in which all the Fourier transform coefficients can be neglected because they decay to zero as frequency tends to infinity, we have

$$\beta_1^2 = DB^2[\bar{g}^2 + \bar{g} + 1] \tag{6.40}$$

Taking the square root we have

$$\beta_1 = B\sqrt{D(\bar{g}^2 + \bar{g} + 1)} \tag{6.41}$$

6.4 Second $k = 2$ Lanczos Iteration derives β_1, x_2, v_2 and α_2

6.4.1 Deriving x_2

$$x_2 = \frac{r_1}{\beta_1} \tag{6.42}$$

Recall $x_1 = \frac{4\sqrt{w_1 w_2}}{\pi}\cos(w_1 x)\cos(w_2 y)$ and $\beta_1 = [DB^2[\bar{g}^2 + \bar{g} + 1]$
and substitute to get

$$= \frac{-4(w_1^2 + w_2^2\cos(w_1 x)\cos(w_2 y)[g(x,y) - B]}{\pi[DB^2[\bar{g}^2 + \bar{g} + 1]}. \tag{6.43}$$

We define a new constant $E = \frac{-4(w_1^2+w_2^2)}{\pi[DB^2[\bar{g}^2+\bar{g}+1]}$
to finish with

$$x_2 = E\cos(w_1 x)\cos(w_2 y)[g(x,y) - B]. \tag{6.44}$$

6.4.2 Deriving v_2

$$v_2 = Lx_2 \tag{6.45}$$

$$= g\Delta x_2 \tag{6.46}$$

$$= g[\frac{d}{dx}\frac{d}{dx}(x_2) + \frac{d}{dy}\frac{d}{dy}(x_2)] \tag{6.47}$$

$$= g[\frac{d}{dx}\frac{d}{dx}(E\cos(w_1 x)\cos(w_2 y)[g(x,y) - B]) + \frac{d}{dy}\frac{d}{dy}(E\cos(w_1 x)\cos(w_2 y)[g(x,y) - B])] \tag{6.48}$$

We separate the derivative into four terms where notation has a 1 or 2 to refer to the first or second term in x_2

$$v_2 = D_{xx1} + D_{xx2} + D_{yy1} + D_{yy2},$$

$D_{xx1} = g[\frac{d}{dx}\frac{d}{dx}(E\cos(w_1 x)\cos(w_2 y)[g(x,y)])])]$
$D_{xx2} = g[\frac{d}{dx}\frac{d}{dx}(E\cos(w_1 x)\cos(w_2 y)[-B]x2)]$
$D_{yy1} = g[\frac{d}{dx}\frac{d}{dx}(E\cos(w_1 x)\cos(w_2 y)[g(x,y)])]$
$D_{yy2} = g[\frac{d}{dx}\frac{d}{dx}(E\cos(w_1 x)\cos(w_2 y)[-B])]$

We move constants to the front before differentiating

$D_{xx1} = Eg\cos(w_2 y)[\frac{d}{dx}\frac{d}{dx}(\cos(w_1 x)g(x,y))]$
$D_{xx2} = -EBg\cos(w_2 y)[\frac{d}{dx}\frac{d}{dx}(\cos(w_1 x))]$
$D_{yy1} = Eg\cos(w_1 x)[\frac{d}{dy}\frac{d}{dy}(\cos(w_2 y)g(x,y))]$
$D_{yy2} = -EBg\cos(w_1 x)[\frac{d}{dy}\frac{d}{dy}(\cos(w_2 y))]$.

Now performing the inner differentiation using the product rule

$$D_{xx1} = Eg\cos(w_2y)\frac{d}{dx}[-(w_1\sin(w_1x)g(x,y)) + \cos(w_1x)g_x(x,y))]$$
$$D_{xx2} = -EBg\cos(w_2y)\frac{d}{dx}[-(w_1\sin(w_1x)]$$
$$D_{yy1} = Eg\cos(w_1x)\frac{d}{dy}[-(w_2\sin(w_2y)g(x,y)) + \cos(w_1x)g_y(x,y)))]$$
$$D_{yy2} = -EBg\cos(w_1x)\frac{d}{dy}[-(w_2\sin(w_2x))].$$

Separating the terms above into six derivative terms and using a notation with a number after the comma to indicate the first or second part in the terms above rule

$$D_{xx1,1} = Eg\cos(w_2y)\frac{d}{dx}[(-w_1\sin(w_1x)g(x,y))]$$
$$D_{xx1,2} = Eg\cos(w_2y)\frac{d}{dx}[\cos(w_1x)g_x(x,y))]$$
$$D_{xx2} = -EBg\cos(w_2y)\frac{d}{dx}[(-w_1\sin(w_1x)]$$
$$D_{yy1,1} = Eg\cos(w_1x)\frac{d}{dy}[(-w_2\sin(w_2y)g(x,y)))]$$
$$D_{yy1,2} = Eg\cos(w_1x)\frac{d}{dy}[\cos(w_1x)g_y(x,y)))]$$
$$D_{yy2} = -EBg\cos(w_1x)\frac{d}{dy}[(-w_2\sin(w_2x))].$$

Then performing the final differentiation we have the following ten derivatives where notation with the letter refers to the first or second term in the product rule

$$D_{xx1,1a} = Eg\cos(w_2y)[(-w_1^2\cos(w_1x)g(x,y)) + (-w_1\sin(w_1x)g_x(x,y))]$$
$$D_{xx1,1b} = Eg\cos(w_2y)[(-w_1^2\cos(w_1x)g(x,y)) + (-w_1\sin(w_1x)g_x(x,y))]$$
$$D_{xx1,2a} = Eg\cos(w_2y)[-w_1\sin(w_1x)g_x(x,y)) + \cos(w_1x)g_{xx}(x,y))]$$
$$D_{xx1,2b} = Eg\cos(w_2y)[-w_1\sin(w_1x)g_x(x,y)) + \cos(w_1x)g_{xy}(x,y))]$$
$$D_{xx2} = -EBg\cos(w_2y)[-w_1^2\cos(w_1x)]$$
$$D_{yy1,1a} = Eg\cos(w_2y)[(-w_2^2\cos(w_2x)g(x,y)) + (-w_1\sin(w_2x)g_y(x,y))]$$
$$D_{yy1,1b} = Eg\cos(w_1x)[(-w_2^2\cos(w_2x)g(x,y)) + (-w_2\sin(w_2x)g_y(x,y))]$$
$$D_{yy1,2a} = Eg\cos(w_1x)[-w_2\sin(w_2x)g_y(x,y)) + \cos(w_2y)g_{yy}(x,y))]$$
$$D_{xx1,2b} = Eg\cos(w_1x)[-w_2\sin(w_2y)g_y(x,y)) + \cos(w_2y)g_{yy}(x,y))]$$
$$D_{yy2} = -EBg\cos(w_1x)[(-w_2^2\cos(w_2y))].$$

Any of the terms above containing sin will be neglected because every w_1x and w_2y are integer multiples of π, for which the evaluation of sin will be 0.

$$D_{xx1,1a} = Eg\cos(w_2y)[(-w_1^2\cos(w_1x)g(x,y))]$$
$$D_{xx1,1b} = Eg\cos(w_2y)[(-w_1^2\cos(w_1x)g(x,y))]$$
$$D_{xx1,2a} = Eg\cos(w_2y)[+\cos(w_1x)g_{xx}(x,y))]$$
$$D_{xx1,2b} = Eg\cos(w_2y)[\cos(w_1x)g_{xy}(x,y))]$$
$$D_{xx2} = -EBg\cos(w_2y)[-w_1^2\cos(w_1x)]$$
$$D_{yy1,1a} = Eg\cos(w_2y)[(-w_2^2\cos(w_2x)g(x,y))]$$
$$D_{yy1,1b} = Eg\cos(w_1x)[(-w_2^2\cos(w_2x)g(x,y))]$$
$$D_{yy1,2a} = Eg\cos(w_1x)[\cos(w_2y)g_{yy}(x,y))]$$
$$D_{xx1,2b} = Eg\cos(w_1x)[\cos(w_2y)g_{yy}(x,y))]$$
$$D_{yy2} = -EBg\cos(w_1x)[(-w_2^2\cos(w_2y))].$$

Combining these final terms we arrive at

$$v_2 = 2E[(\frac{B}{2}-1)(w_1^2+w_2^2)+g_{xx}(x,y)+g_{yy}(x,y)]g(x,y)\cos(w_1x)\cos(w_2y). \tag{6.49}$$

We define a constant $F = (\frac{B}{2}-1)(w_1^2+w_2^2)$ so that we finally have

$$v_2 = 2E[F+g_{xx}(x,y)+g_{yy}(x,y)]g(x,y)\cos(w_1x)\cos(w_2y). \tag{6.50}$$

6.4.3 Deriving α_2

$$\alpha_2 = \langle x_2, v_2\rangle \tag{6.51}$$

Recall $x_2 = E\cos(w_1x)\cos(w_2y)[g(x,y)-B]$,
$v_2 = 2E[F+g_{xx}(x,y)+g_{yy}(x,y)]g(x,y)\cos(w_1x)\cos(w_2y)$,
where the constant E is defined as $E = \frac{-4(w_1^2+w_2^2)}{\pi[DB^2[\bar{g}^2+\bar{g}+1]}$,
and the inner product $\langle x_2, v_2\rangle = \int_0^{pi}\int_0^{pi} x_2v_2dydx$.

$$\alpha_2 = \int_0^{\pi}\int_0^{\pi} x_2v_2dydx \tag{6.52}$$

$$= \int_0^{\pi}\int_0^{\pi} [E\cos(w_1x)\cos(w_2y)[g(x,y)-B]][2E[F+g_{xx}(x,y)+g_{yy}(x,y)]g(x,y)\cos(w_1x)\cos(w_2y)]]dydx \tag{6.53}$$

We define a new constant H such that $\frac{H}{\pi} = E$ so $\frac{H}{\pi} = \frac{-4(w_1^2+w_2^2)}{[DB^2[\bar{g}^2+\bar{g}+1]}$
which allows us to can keep the π terms visible in α_2 so later in the derivation we can recognize the Fourier transform coefficient form.

$$= \int_0^{\pi}\int_0^{\pi} [\frac{H}{\pi}\cos(w_1x)\cos(w_2y)[g(x,y)-B]][2\frac{H}{\pi}[F+g_{xx}(x,y)+g_{yy}(x,y)]g(x,y)\cos(w_1x)\cos(w_2y)dydx \tag{6.54}$$

Now we combine constants carry out the product to get six integrals
$\alpha_2 = I_1+I_2+I_3+I_4+I_5+I_6$

$I_1 = 2FH^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi} g^2(x,y)\cos^2(w_1x)\cos^2(w_2y)dydx$
$I_2 = 2H^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi} g_{xx}(x,y)g^2(x,y)\cos^2(w_1x)\cos^2(w_2y)dydx$
$I_3 = 2H^2\frac{1}{\pi^2}+\int_0^{\pi}\int_0^{\pi} g_{yy}(x,y)g^2(x,y)\cos^2(w_1x)\cos*2(w_2y)dydx$
$I_4 = -2BFH^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi}\cos^2(w_1x)\cos^2(w_2y)dydx$
$I_5 = -2BH^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi} g_{xx}(x,y)g(x,y)\cos^2(w_1x)\cos^2(w_2y)dydx$
$I_6 = -2BH^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi} g_{yy}(x,y)g(x,y)\cos^2(w_1x)\cos^2(w_2y)dydx.$

Using the half angle formula twice, the product $\cos^2(w_1x)\cos^2(w_2y) = [\frac{1}{2}(\cos 2(w_1x)+1][\frac{1}{2}(\cos 2(w_2y)+1]$
which when expanded has the terms $\frac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)+\cos 2(w_1x)+\cos 2(w_2y)+1]$
as follows,

$I_1 = 2FH^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi} g^2(x,y)\frac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)+\cos 2(w_1x)+\cos 2(w_2y)+1]dydx$
$I_2 = 2H^2\frac{1}{\pi^2}\int_0^{\pi}\int_0^{\pi} g_{xx}(x,y)g^2(x,y)\frac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)+\cos 2(w_1x)+\cos 2(w_2y)+1]dydx$

$$I_3 = 2H^2\tfrac{1}{\pi^2} + \int_0^\pi \int_0^\pi g_{yy}(x,y)g^2(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y) + \cos 2(w_1x) + \cos 2(w_2y) + 1]dydx$$
$$I_4 = -2BFH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y) + \cos 2(w_1x) + \cos 2(w_2y) + 1]dydx$$
$$I_5 = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y) + \cos 2(w_1x) + \cos 2(w_2y) + 1]dydx$$
$$I_6 = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{yy}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y) + \cos 2(w_1x) + \cos 2(w_2y) + 1]dydx.$$

Separating the integrals into 4 subgroups for each term in the expansion of the product of the half angle formulas we have

$$I_{1a} = 2FH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g^2(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)]dydx$$
$$I_{2a} = 2H^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g^2(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)]dydx$$
$$I_{3a} = 2H^2\tfrac{1}{\pi^2} + \int_0^\pi \int_0^\pi)g_{yy}(x,y)g^2(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)]dydx$$
$$I_{4a} = -2BFH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)]dydx$$
$$I_{5a} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)]dydx$$
$$I_{6a} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{yy}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_1x)\cos 2(w_2y)]dydx$$
$$I_{1b} = 2FH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g^2(x,y)\tfrac{1}{4}[\cos 2(w_1x)]dydx$$
$$I_{2b} = 2H^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g^3(x,y)\tfrac{1}{4}[\cos 2(w_1x)]dydx$$
$$I_{3b} = 2H^2\tfrac{1}{\pi^2} + \int_0^\pi \int_0^\pi g_{yy}(x,y)g^3(x,y)\tfrac{1}{4}[\cos 2(w_1x)]dydx$$
$$I_{4b} = -2BFH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g(x,y)\tfrac{1}{4}[\cos 2(w_1x)]dydx$$
$$I_{5b} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_1x)]dydx$$
$$I_{6b} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{yy}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_1x)]dydx$$
$$I_{1c} = 2FH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g^2(x,y)\tfrac{1}{4}[\cos 2(w_2y)]dydx$$
$$I_{2c} = 2H^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g^2(x,y)\tfrac{1}{4}[\cos 2(w_2y)]dydx$$
$$I_{3c} = 2H^2\tfrac{1}{\pi^2} + \int_0^\pi \int_0^\pi g^2(x,y)\tfrac{1}{4}[\cos 2(w_2y)]dydx$$
$$I_{4c} = -2BFH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g(x,y)\tfrac{1}{4}[\cos 2(w_2y)]dydx$$
$$I_{5c} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_2y)]dydx$$
$$I_{6c} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{yy}(x,y)g(x,y)\tfrac{1}{4}[\cos 2(w_2y)]dydx$$
$$I_{1d} = 2FH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g^2(x,y)\tfrac{1}{4}dydx$$
$$I_{2d} = 2H^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g^2(x,y)\tfrac{1}{4}dydx$$
$$I_{3d} = 2H^2\tfrac{1}{\pi^2} + \int_0^\pi \int_0^\pi g_{yy}(x,y)g^2(x,y)\tfrac{1}{4}dydx$$
$$I_{4d} = -2BFH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g(x,y)\tfrac{1}{4}dydx$$
$$I_{5d} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{xx}(x,y)g(x,y)\tfrac{1}{4}dydx$$
$$I_{6d} = -2BH^2\tfrac{1}{\pi^2}\int_0^\pi \int_0^\pi g_{yy}(x,y)g(x,y)\tfrac{1}{4}dydx.$$

Moving the $\frac{1}{4}$ constant to the front and recognizing the integral forms to be Fourier cosine coefficients for the first 18 integrals we have the following

$$I_{1a} = \tfrac{FH^2}{2}\widehat{g^2}(w_1,w_2)$$
$$I_{2a} = \tfrac{H^2}{2}\widehat{g_{xx}g^2}(w_1,w_2)$$
$$I_{3a} = \tfrac{H^2}{2}\widehat{g_{yy}g^2}(w_1,w_2)$$
$$I_{4a} = -\tfrac{BFH^2}{2}\hat{g}(w_1,w_2)$$
$$I_{5a} = -\tfrac{BH^2}{2}\widehat{g_{xx}g}(w_1,w_2)$$
$$I_{6a} = -\tfrac{BH^2}{2}\widehat{g_{yy}g}(w_1,w_2).$$

The b integrals having only $\cos(2w_1x)$ have $w_2 = 0$

$I_{1b} = \frac{FH^2}{2}\widehat{g^2}(w_1,0)$
$I_{ba} = \frac{H^2}{2}\widehat{g_{xx}g^2}(w_1,0)$
$I_{3b} = \frac{H^2}{2}\widehat{g_{yy}g^2}(w_1,0)$
$I_{4b} = -\frac{BFH^2}{2}\hat{g}(w_1,0)$
$I_{5b} = -\frac{BH^2}{2}\widehat{g_{xx}g}(w_1,0)$
$I_{6b} = -\frac{BH^2}{2}\widehat{g_{yy}g}(w_1,0).$

The c integrals having only $\cos(2w_2x)$ have $w_1 = 0$
$I_{1c} = \frac{FH^2}{2}\widehat{g}(0,w_2)$
$I_{2c} = \frac{H^2}{2}\widehat{g_{xx}g^2}(0,w_2)$
$I_{3c} = \frac{H^2}{2}\widehat{g_{yy}g^2}(0,w_2)$
$I_{4c} = -\frac{BFH^2}{2}\hat{g}(0,w_2)$
$I_{5c} = -\frac{BH^2}{2}\widehat{g_{xx}g}(0,w_2)$
$I_{6c} = -\frac{BH^2}{2}\widehat{g_{yy}g}(0,w_2).$

The d integrals have no cos terms and evaluate to average values on the integral

$I_{1d} = \frac{FH^2}{2}\bar{g}$
$I_{2d} = \frac{H^2}{2}\bar{g}_{xx}\bar{g}^2$
$I_{3d} = \frac{H^2}{2}\bar{g}_{yy}\bar{g}^2$
$I_{4d} = -\frac{BFH^2}{2}\bar{g}$
$I_{5d} = -\frac{BH^2}{2}\bar{g}_{xx}\bar{g}$
$I_{6d} = -\frac{BH^2}{2}\bar{g}_{yy}\bar{g}.$

As described in Section 2.10.2, all Fourier transform coefficients tend to 0 as w_1, and w_2 tend to infinity, which allows for neglecting the first 18 integrals, because they are various Fourier transform coefficients. This simplifies to allow for six integrals to remain and α_2 becomes,

$$\alpha_2 = \frac{FH^2}{2}\bar{g} + \frac{H^2}{2}\bar{g}_{xx}\bar{g}^2 + \frac{H^2}{2}\bar{g}_{yy}\bar{g}^2 - \frac{BFH^2}{2}\bar{g} - \frac{BH^2}{2}\bar{g}_{xx}\bar{g} - \frac{BH^2}{2}\bar{g}_{yy}\bar{g}. \tag{6.55}$$

Factoring out constants we get a simplified form

$$\frac{H^2}{2}[F\bar{g} + \bar{g}_{xx}\bar{g}^2 + \bar{g}_{yy}\bar{g}^2 - BF\bar{g} - B\bar{g}_{xx}\bar{g} - B\bar{g}_{yy}\bar{g}] \tag{6.56}$$

$$\alpha_2 = \frac{H^2}{2}[(\bar{g}^2 - B\bar{g})(F + \bar{g}_{xx} + \bar{g}_{yy})]. \tag{6.57}$$

This completes the derivation of the Lanzcos recursion coefficients.