The Effect of Image Preprocessing Techniques and Varying JPEG Quality on the Identifiability of Digital Image Splicing Forgery

by

Aaron Gubrud

A Thesis Presented in Partial Fulfillment
of the Requirements for the Degree
Master of Science

Approved April 2015 by the
Graduate Supervisory Committee:

Baoxin Li, Chair
K. Selçuk Candan
Zafer Kadi

ARIZONA STATE UNIVERSITY

May 2015

ABSTRACT

Splicing of digital images is a powerful form of tampering which transports regions of an image to create a composite image. When used as an artistic tool, this practice is harmless but when these composite images can be used to create political associations or are submitted as evidence in the judicial system they become more impactful. In these cases, distinction between an authentic image and a tampered image can become important.

Many proposed approaches to image splicing detection follow the model of extracting features from an authentic and tampered dataset and then classifying them using machine learning with the goal of optimizing classification accuracy. This thesis approaches splicing detection from a slightly different perspective by choosing a modern splicing detection framework and examining a variety of preprocessing techniques along with their effect on classification accuracy. Preprocessing techniques explored include Joint Picture Experts Group (JPEG) file type block line blurring, image level blurring, and image level sharpening. Attention is also paid to preprocessing images adaptively based on the amount of higher frequency content they contain.

This thesis also recognizes an identified problem with using a popular tampering evaluation dataset where a mismatch in the number of JPEG processing iterations between the authentic and tampered set creates an unfair statistical bias, leading to higher detection rates. Many modern approaches do not acknowledge this issue but this thesis applies a quality factor equalization technique to reduce this bias. Additionally, this thesis artificially inserts a mismatch in JPEG processing iterations by varying amounts to determine its effect on detection rates.

ACKNOWLEDGEMENTS

As much as the completion of a thesis is an indication of the drive and determination of the person attached to the byline there is an implicit statement about the quality of people who have supported them in their journey. I believe this endeavor would have been at least tenfold more difficult, if not impossible, in their absence.

I would first like to thank Dr. Baoxin Li for his support and guidance throughout this entire project. Having access to someone with such a depth and breadth of image processing knowledge was immensely valuable while conducting my research and forming my thesis.

I also would like to thank Dr. K. Selçuk Candan and Dr. Zafer Kadi for serving as members on my committee and evaluating my thesis. Each of them are also responsible for introducing me to new and interesting applications of video and image processing which ultimately led to me selecting this as my field of research.

Thanks to Parag Chandakkar for introducing me to the field of image splicing detection and for helping me establish my foundation in this focus. His patience and explanation skills made a challenging topic more digestible. It was with the concepts you helped me understand that I was able to venture out and make this project my own.

Finally, I would like to thank family, friends, and coworkers who have tolerated my limited free time and thesis-induced moodiness throughout the last year and a half. Your support and encouragement were invaluable in keeping me motivated to see this through to the finish. These sentences feel shamefully disproportionate to the degree of gratitude that I feel and yet no number of words could hope to express it fully.

Table of Contents

CHAPTER Page

1 INTRODUCTION 1

1.1 Problem Statement 1

1.2 Objectives 2

1.3 Methodology 3

1.4 Contributions 4

2 BACKGROUND / ANALYSIS OF SPLICING DETECTION 5

2.1 Detection Techniques 6

2.1.1 Active Detection Techniques 7

2.1.2 Passive Detection Techniques 9

2.2 Image Tampering Datasets 13

2.2.1 Columbia Image Splicing Detection Evaluation Dataset 13

2.2.2 Columbia Uncompressed Image Splicing Detection Evaluation Dataset 14

2.2.3 CASIA Tampered Image Detection Evaluation (TIDE) Database V1 15

2.2.4 CASIA Tampered Image Detection Evaluation (TIDE) Database V2 16

2.2.5 The Importance of Equalizing JPEG Compression Applications 17

3 PROPOSED FRAMEWORK 20

3.1 Framework Overview 20

CHAPTER Page

3.2 Selection of Images for Testing 21

3.3 Preprocessing of Images 21

3.3.1 JPEG Quality Factor Equalization 22

3.3.2 Color Space Conversion 24

3.3.3 JPEG Block Line Blurring 26

3.3.4 Image Level Filtering 31

3.3.5 Content Adaptive Techniques 39

3.4 Feature Extraction 44

3.5 Classification 46

4 ALGORITHM IMPROVEMENTS AND QUALITY FACTOR MISMATCH EFFECT 51

4.1 Experimental Results for Preprocessing Techniques 52

4.1.1 Experimental Results for Color Space Conversion 52

4.1.2 Experimental Results for JPEG Block Line Blurring 52

4.1.3 Experimental Results for Image Level Filtering 53

4.1.4 Experimental Results for Content Adaptive Techniques 58

4.1.5 Experimental Results for Combining Preprocessing Techniques 63

4.2 Classifier Bias Towards Identifying Tampered Images 66

CHAPTER Page

4.3 Experimental Results for Divergent Quality Factor Detection 70

4.3.1 Effect of Divergent Quality Factor Detection..... 73

5 CONCLUSION 75

6 REFERENCES 77

List of Tables

Table Page

2.1: Comparison Of Edge Image Statistics [18] 19

3.1: Performance of Reference Algorithm on Un-Equalized Quality Factors 22

3.2: Performance Of Reference Algorithm On Dataset With Equalized Quality Factors 23

3.3: MATLAB Blurring Filter Types and Parameters 32

3.4: MATLAB Image Sharpening Filter and Parameters 37

4.1: Reference Algorithm Performed In RGB And YCbCr Color Space 52

4.2: Results For JPEG Block Blur With Various Weights And Neighborhoods 53

4.3: Results For Modifying Size Of Averaging Filter 54

4.4: Results For Image Level Gaussian Blurring With Various Parameters 54

4.5: Results For Image Level Sharpening With Various Amounts 57

4.6: Percentile Mappings To Threshold μ Values 59

4.7: Results For Blurring Smoother Images More Versus Less 59

4.8: Results For Choosing Different μ Thresholds 60

4.9: Results For Smaller Spans Of σ 61

4.10: Results For Linearly Determined σ With Varying σ Spans 62

Table Page

4.11: Results For σ Span 0.5-0.625 With Varying JPEG Block Line Blurring Configurations 64

4.12: Results For σ Span 0.625-0.75 With Varying JPEG Block Line Blurring Configurations 64

4.13: Results For σ Span 0.5-0.75 With Varying JPEG Block Line Blurring Configurations 65

4.14: Results For Keeping JPEG Block Line Blurring Constant With Varying Constant σ Values 65

4.16: Results For Different Quality Factor Mismatches 72

4.17: Statistics For Varying Quality Factors 73

4.18: Results For Top Performing Approaches Using Un-Equalized CASIA TIDE Database V2 73

4.19: Statistics For Varying Quality Factors (Worst Case Omitted) 74

List of Figures

Figure Page

2.1: Duplicating A Region And Repositioning It Within The Same Image [1] 5

2.2: Extracting A Region From A Source Image And Moving It Into A Destination Image [1] 5

2.3: Controversial Image Splicing Forgery Combining Two Different Images For Political Means [3] 6

2.4: Block Diagram Of A Typical Digital Image Acquisition Pipeline 8

2.5: Example Authentic (Left) And Tampered (Right) Images In The Columbia "Image Splicing Detection Evaluation Dataset" 14

2.6: Example Authentic (Left) And Tampered (Right) Images In The Columbia "Uncompressed Image Splicing Detection Evaluation Dataset" 15

2.7: Example Authentic (Left) And Tampered (Right) Images From The CASIA TIDE Database V1 16

2.8: Example Authentic (Left) And Tampered (Right) Images From The CASIA TIDE Database V2 17

3.1: JPEG Processing Block Diagram [52] 27

3.2: JPEG-Compressed Image With Evident Block Lines (Au_Nat_00093.jpg) 28

Figure Page

3.3: Zoomed-In Region Of JPEG-Compressed Image With Evident Block Lines (Au_Nat_00093.jpg) 28

3.4: Au_Nat_00093.Jpg After Block Line Blurring 29

3.5: Zoomed-In Region Of Au_Nat_00093.jpg After Block Line Blurring 29

3.6: Illustration Of Block Line Blurring With Sample Coefficients 30

3.7: Impact Of Changing JPEG Block Line Blurring Weightings On Frequency Content 31

3.8: Au_ani_10208.jpg Original (left) And Gaussian Blurred With $\sigma = 0.5$ And $hsize = 3x3$ (right) 33

3.9: Frequency Content Of Au_Ani_10208.jpg (Original) 34

3.10: Frequency Content Of Au_Ani_10208.jpg After Gaussian Blur 34

3.11: Impact of Changing Gaussian Blurring Filter Parameters On Frequency Content . 35

3.12: Impact of Changing Averaging Blurring Filter Parameters On Frequency Content 36

3.13: Au_Ani_10208.jpg Original (Left) And Sharpened With Amount = 0.75 (Right) . 38

3.14: Impact Of Changing Sharpening Filter Parameters On Frequency Content 38

3.15: Illustration Of DCT Frequency Ordering [56] 39

3.16: Zigzag Pattern Followed By DCT Coefficients [56] 40

Figure Page

3.17: Scatterplot Of Quality Factor Equalized CASIA TIDE V2 Authentic Average DCT Coefficients (μ) 41

3.18: Scatterplot Of Quality Factor Equalized CASIA TIDE V2 Tampered Average DCT Coefficients (μ) 41

3.19: Image With Lowest μ (Left, $\mu = 0.458$) And Highest μ (Right, $\mu = 33.57$) From The CASIA TIDE Database V2 Authentic Set 42

3.20: 2D Projection Of Features With Two Classes [58] 47

3.21: Classes That Are Not Linearly Separable [58] 48

4.1: Averaging Filter On Au_Ani_10208.jpg (Top Left) With $hsize = 3 \times 3$ (Top Right), $hsize = 5 \times 5$ (Lower Left), And $hsize = W \times H$ (Lower Right) 55

4.2: Gaussian Filter On Au_Ani_10208.jpg (Left) With $hsize = W \times H$, $\sigma = 1.0$ (Right) 56

4.3: Improper Classification Rates In Top 80th Percentile 68

4.4: Improper Classification Rates In Top 90th Percentile 68

4.5: Improper Classification Rates In Top 90th Percentile 69

4.6: Improper Classification Rates In Top 99th Percentile 69

4.7: Project Structure Thus Far 71

4.8: Testing Structure For This Section 71

1 INTRODUCTION

This chapter provides the problem statement addressed in this thesis, objectives this thesis hopes to achieve, the methodology followed, and contributions made to the field of image splicing detection.

1.1 PROBLEM STATEMENT

In the last few decades, generation and consumption of digital media has seen a dramatic increase around the world. Coupled with the growth of consumer compute power, evolving software products, and a global sharing mechanism (i.e. the Internet and content hosting sites), it is now easier than ever to create, manipulate and share content. While the evolution of techniques and software surrounding digital media have allowed users to express themselves in interesting and explorative ways, it has also become increasingly difficult to determine the authenticity of content when a user encounters it blindly; particularly in the case of digital imaging. While musings on the authenticity of an image can be trivial, the distinction between authentic and tampered content can also have more serious implications. In the realms of politics and the judicial system, for instance, manipulation of images can sway opinion of a public figure or influence a verdict in a court of law if they are perceived to be authentic. An image can be tampered in a number of different ways, but image splicing is arguably the most powerful tampering operation. Image splicing means taking a region from a source image and transporting it to a destination image. The source and destination image can either be the same or different but regardless it can have a substantial impact on the high level message of an image. There

have been a number of splicing detection frameworks proposed over the years and though a number of them have produced impressively accurate solutions, a subtle characteristic of the datasets used to test them may be unfairly biasing their reported accuracy. Because a tampered image may contain a region from a JPEG-processed image and then processed and written as a JPEG, tampered images often have two iterations of JPEG processing applied to them. Authentic images, on the other hand, will only have one iteration of JPEG processing applied to them. This mismatch in the number of iterations of JPEG processing reveals itself in a statistical pattern that makes differentiating authentic and tampered images more simple than can be expected with real world images. Additionally, with so much focus on identifying novel splicing detection frameworks, the value of preprocessing images before splicing detection has gone largely unexplored.

1.2 OBJECTIVES

This thesis aims to further the research field of image splicing detection through the following explorations:

1. Examine the effect of various preprocessing techniques on the quality of feature vectors extracted for the purpose of differentiating authentic images from images which contain spliced regions
2. Vary the JPEG quality factor levels between authentic and tampered sets and examine the impact on detection accuracy rates

1.3 METHODOLOGY

Many image splicing detection publications are concerned with uncovering new methods of feature extraction that produce vectors which can accurately lead to the identification of an authentic or tampered image. This study is unique in that it uses an established splicing detection framework and instead focuses on a variety of image preprocessing techniques that look to enhance accuracy rates. Preprocessing techniques include blurring along JPEG block line boundaries and image level filtering. Content adaptive preprocessing techniques are also explored.

Once feature vectors have been extracted from every image in the authentic and tampered set, they are used to train a machine learning classifier. Another set of vectors test the machine learning classifier, thus evaluating the effectiveness of the preprocessing techniques explored.

Research in the field of image splicing detection calls attention to the impact of a mismatch in JPEG quality factors between authentic and tampered image datasets. This mismatch has been shown to influence classification through statistical patterns which effectively makes splicing detection easier. This approach effectively equalizes the quality factors for each dataset in order to neutralize this statistical bias and test detection techniques in a more realistic way.

The final section of this thesis takes a more in-depth look at the effect of mismatches in quality factor between the authentic and tampered sets. Whereas the goal in previous sections of this thesis is to evaluate performance while both sets are on equal

footing, this section gradually varies an artificially created quality factor mismatch to examine its effect on detection rates.

1.4 CONTRIBUTIONS

This thesis shifts focus from identifying new feature extraction methods and instead proposes that the quality of features extracted from existing techniques can be increased. Its focus on preprocessing techniques has not been encountered in any other publications that are currently available. Moreover by exploring the impact of varying JPEG quality factors more thoroughly, it advances research awareness of how to handle content with possibly unknown levels of compression when encountered in the real world.

2 BACKGROUND / ANALYSIS OF SPLICING DETECTION

Anyone who has sat down to use an image editing software will recognize that there are many ways to manipulate an image. Of these manipulation possibilities, none have quite as much power to change the high level message of an image as splicing. Splicing copies a region from a source image and pastes it into a destination image. In some cases, the source and destination image are the same one (*Figure 2.1*) but they can also be different (*Figure 2.2*).

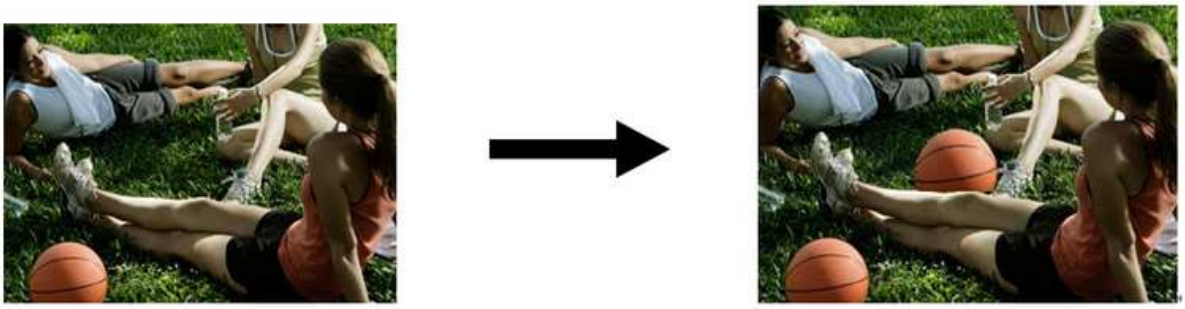

Figure 2.1: Duplicating A Region And Repositioning It Within The Same Image [1]

Figure 2.2: Extracting A Region From A Source Image And Moving It Into A Destination Image [1]

While the above examples are harmless, splicing forgeries can have powerful messages such as in the doctored image below (*Figure 2.3*) which surfaced during the 2004

United States Presidential Election and was a forgery that some believe intended to "embarrass the Democratic frontrunner, John Kerry" [2].

Figure 2.3: Controversial Image Splicing Forgery Combining Two Different Images For Political Means [3]

Figure 2.3 and its associated controversy [4] illustrates the point that the authenticity of an image can be a nontrivial attribute. For this reason, there is an understandable need for mechanisms which can accurately identify an image that has been spliced.

2.1 DETECTION TECHNIQUES

Querying a technical publication database such as IEEE Xplore [5] will show that the topic of "image tampering detection" has been researched in a traceable way since 1998 [6]. Looking at the titles of early papers (submitted between 1998 and 2000), the prevalence of the keyword "watermarking" indicates the frame of mind with which image tampering detection was first approached. Similar to the concept of watermarking in the physical world, watermarking for digital images embeds a verifiable piece of information into the image that serves as an indicator of authenticity or origin. This type of tampering

prevention is reliant on “active detection techniques” as opposed to “passive detection techniques”. The distinction between these two terms will be explained below.

2.1.1 Active Detection Techniques

Active detection techniques operate on the principle that there is some sort of data structure embedded within an image that vouches for its authenticity or origin. An example of this is watermarking, as was mentioned earlier. In the digital realm watermarking is an application of digital steganography which is the concept of hiding data within data [7]. Digital mediums that are tolerant to noise (e.g. images and audio) are candidates for steganographic techniques such as watermarking because additional data can be almost imperceptibly added to the carrier – especially when added to frequency representations of the signal.

Much like cryptographically signing a digital package, however, it is important to have a trusted entity that is certifying a particular package. In the same way that website certificates are only as credible as the certificate authority, the credibility of a watermarked image can depend on the entity that applies the watermark. Because of the data flow associated with digital image acquisition, the step in the process where watermarking is applied can also affect how credible an image’s authenticity is. [8] provides a high level block diagram of a typical pipeline that a digital image capture follows:

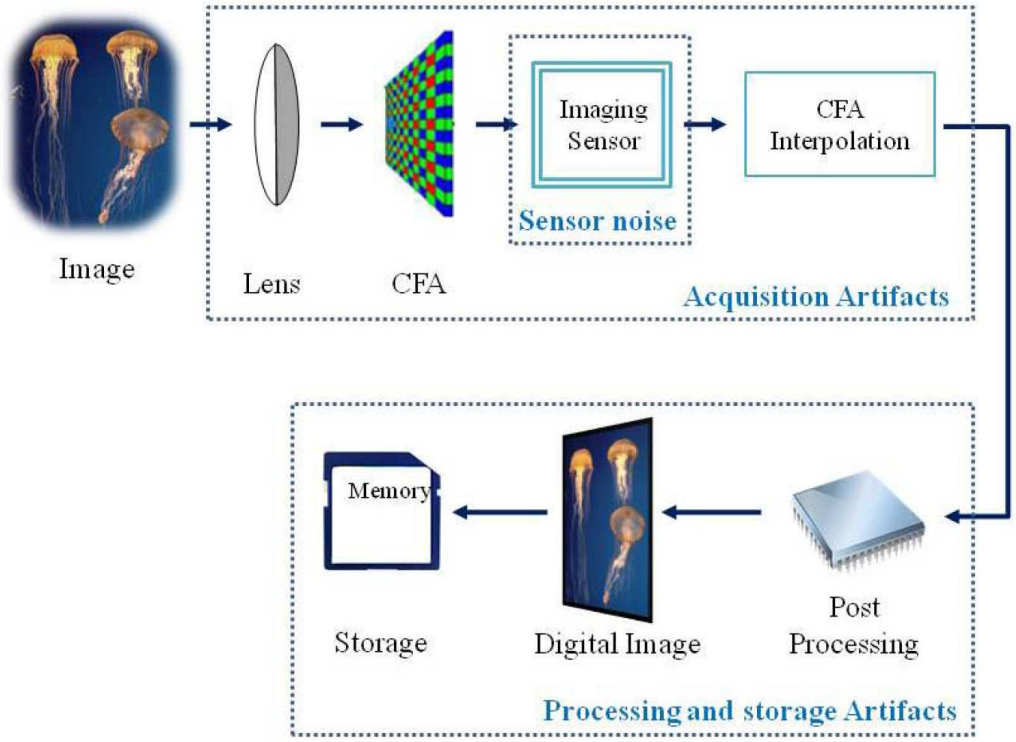

Figure 2.4: Block Diagram Of A Typical Digital Image Acquisition Pipeline

As *Figure 2.4* illustrates, there are numerous steps in the pipeline where a watermark could be embedded in an image. Proposals tend to apply watermarking at the end of the pipeline – either between the digital image and storage or after the data has reached storage.

The former option would require watermarking hardware to be included in the camera hardware itself such as in [9]. This is advantageous because it enables a camera to self-authenticate the images it captures before it makes its way to storage or even leaves the device. Researches such as in [10] also acknowledge the need of a real time solution to camera-contained hardware watermarking as the image taking process should not be gated

by processing time to perform the watermarking operations. As of the time of writing, there are no commercially available cameras that watermark automatically within a device. [9] identifies models made by manufacturers Kodak and Epson but even at the time of its publishing (2004) these models were already discontinued.

If watermarking is not to be done within the camera device itself, then that means it is to be done after the digital image already makes it to the storage phase. Methods targeting this step can be inherently less trustworthy because there is an infinite amount of time where the watermarking process can be applied after the storage stage has been reached. This means that an image could be captured, stored, manipulated, and then passed off as authentic.

Regardless of which stage a watermark is applied, its "fragility" can lend to the ability of detecting tampering [11]. In some implementations, fragile watermarking techniques can not only tell if an image has been tampered but which regions have been attacked. Fragile watermarks do have their challenges, however, as innocent operations such as lossy compression can also destroy them.

2.1.2 Passive Detection Techniques

In an ideal world, active detection techniques would be an effective and reliable indicator of whether an image has been tampered. However, their limitations call for techniques that blindly evaluate authenticity. Passive detection techniques make informed guesses about whether an image has been tampered based on exploiting patterns across both authentic and tampered content. Contrasting the watermark-centric approach that dominated tampering research nearly two decades ago, most modern approaches share a

common element of extracting feature vectors and classifying them with machine learning. These passive detection techniques often bank on using machine learning frameworks (such as libSVM [12]) to train a classifier on feature vectors that indicate if an image is tampered or authentic and then testing the classifier to see if it properly categorizes an arbitrary set. The high level process for these modern approaches is fairly simple:

1. Reference a dataset of authentic images and a dataset of tampered images
2. For each image in the authentic dataset, extract a meaningful feature vector and create a data structure of these feature vectors for the entire authentic dataset
3. For each image in the tampered dataset, extract a meaningful feature vector and create a data structure of these feature vectors for the entire tampered dataset
4. Train the classifier with examples of feature vectors from authentic images and examples of feature vectors from tampered images
5. Test the classifier by presenting it with unmarked authentic images and unmarked tampered images and record the rate at which authentic images are identified as authentic (true positive), tampered images are identified as tampered (true negative), authentic images are identified as tampered (false negative) and tampered images are identified as authentic (false positive).

[13] surveys passive image splicing detection techniques that have emerged between 2004-2015, organizing the described frameworks into those which examine image data as standalone data structure and those which attempt to correlate an image back to its capture device or to the environment in which it was captured. The next two sections explore these branches of passive detection.

2.1.2.1 Passive Techniques Treating Images As A Standalone Entity

A number of approaches successfully use image data by itself to draw conclusions about image authenticity. One subset of these techniques leverage the mindset that an image is a two dimensional signal. With this focus an analysis of resampling/interpolation inconsistencies ([14] , [15]) or noise inconsistencies ([16]) can be used to indicate splicing forgery. [17] uses bicoherence analysis which traditionally has applications in human speech splicing detection and aims to catch "abrupt discontinuities in an image" inherent to splicing [13]. [18] looks to frequency analysis of multiple block sizes with DCT and transitions between these blocks (Markovian rake). [19] uses the "statistical difference between images and its filter version, ten metrics including absolute error, mean square error, four correlation measures, two spectral measures and two human visual system measures" [13].

Other techniques look to image structure to identify splicing attempts. This is to say that they focus on texture analysis, pixel adjacency or segmentation of distinct regions for example. [20] and [21] use texture analysis through local binary pattern and multi resolution Weber descriptors respectively. [22] leverages probabilities of transitions between adjacent pixels using a Markov chain while [23] improves this approach by using more complex sets of pixel neighbors. [24] segments images and analyzes region-based prediction error in combination with Haralick texture features. [25] examines high level image structure in the context of saliency and proposes saliency features augment existing tampering detection methods.

One final subset of these techniques looks to the problem from the mindset of the splicing process and high level patterns that will be left behind; particularly steps taken to conceal the tampering effort. [26] aims to distinguish manual blurring from natural blurring by modeling the Gaussian blur radius along edges while [27] and [28] examine the problem from an edge width perspective. [29] also looks to model manual blurring but expands the effort to group regions with similar blur models. [30] estimates the total amount of blurring on a block-by-block basis to elicit inconsistent blur amounts. [31] examines image gradients to model motion blurring in an image and identify when an individual has attempted to conceal a splice with this type of blur.

2.1.2.2 Passive Techniques Treating Images As A Product Of The Environment

An alternative to evaluating images as self-contained entities is to consider the context in which they were captured. All authentic images were captured by an imaging device located in some environment and these techniques aim to model both of these components to identify inconsistencies.

Approaches characterizing the camera with which a picture was taken identify inconsistencies in regions that point to different capture devices. [32] examines artifacts induced by demosaicing. [33] looks at the sharpness of colors and approximates what a demosaiced image should look like to differentiate between authentic and tampered images. [34] leverages the combination of a camera's unique combination of a CCD/CMOS sensor, demosaicing filter, and camera response function to generate a fingerprint to assign to image regions. [35] attempts to profile 17 different CCDs to map

image regions to capture sensors. [36] analyzes geometric features in an image to associate with a specific capture device and use inconsistencies to point to tampering attempts.

Some other approaches look to identify splicing attempts through inconsistencies in the ambient environment of the scene. [37] exclusively examines images where people are the subjects and models an illuminant color to identify inconsistencies. [38] also looks to illuminant inconsistency but does so with all types of subjects.

2.2 IMAGE TAMPERING DATASETS

As can be seen in the previous section, there are an abundance of passive detection techniques; some with very specific focuses. Without a standard dataset of authentic and tampered images, it would be difficult to conduct any sort of meaningful comparison between competing implementations of the solution to this problem. This is because digital images are extremely diverse; the same can be said of tampering techniques. To support research of splicing detection, multiple image datasets comprised of authentic and tampered images have been generated. The most well-known datasets originate from Columbia University and the Institute of Automation, Chinese Academy of Sciences (CASIA).

2.2.1 Columbia Image Splicing Detection Evaluation Dataset

Columbia University was the first to introduce a dataset to further research in this field with their "Image Splicing Detection Evaluation Dataset" in 2004 [39]. This dataset consists of authentic and spliced image blocks, which are further divided into different subcategories (smooth vs. textured, arbitrary object boundary vs. straight boundary)" [39].

It has 933 authentic images and 912 tampered images. All of the authentic and tampered images are 128x128 resolution, have no color information, and are an uncompressed file format (BMP). Images in the “Image Splicing Detection Evaluation Dataset” are not full compositions but instead regions from images in the CalPhotos collection [40].

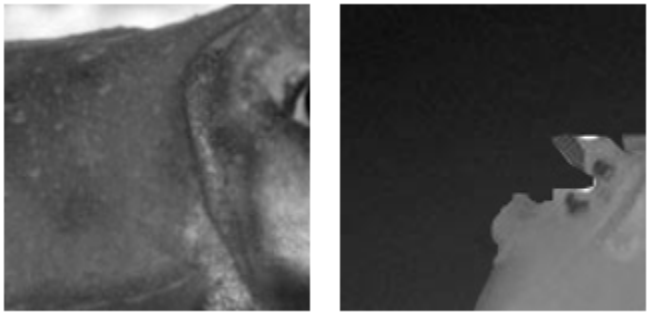

Figure 2.5: Example Authentic (Left) And Tampered (Right) Images In The Columbia "Image Splicing Detection Evaluation Dataset"

2.2.2 Columbia Uncompressed Image Splicing Detection Evaluation Dataset

In 2006, Columbia University released another dataset called “Columbia Uncompressed Image Splicing Detection Evaluation Dataset” [41]. In many ways this dataset is more robust than their initial offering, with images of varying resolutions (757x568 to 1152x768) that contain color information, and again are stored in an uncompressed file format (TIF). Spliced regions are incorporated without any post processing and visually most of the spliced regions can be readily identified with the naked eye. Images in this dataset were captured by the researchers themselves and include 183 authentic images and 180 tampered images. This set is an improvement over the first set

because the inclusion of content with color information and full compositions more closely mimics images that the average person could encounter.

Figure 2.6: Example Authentic (Left) And Tampered (Right) Images In The Columbia "Uncompressed Image Splicing Detection Evaluation Dataset"

2.2.3 CASIA Tampered Image Detection Evaluation (TIDE) Database V1

With these two datasets already established, CASIA developed their own dataset of authentic and tampered images called the TIDE database v1 [42] in 2010. The CASIA TIDE database v1 is composed of 800 authentic and 925 tampered images. All of the authentic and tampered images are 384x256 pixel resolution with color information and in a compressed file format (JPEG). CASIA TIDE v1 makes a commitment to providing a diverse set of authentic and tampered images:

- 8 different subject categories (Scene, Animal, Architecture, Character, Article, Plant, Nature, Texture)
- Classification for tampered region type (spliced region from within same image or from different image)

- Splicing operation type (spliced region is resized, deformed, rotated, or left alone)

This set features images from the Corel dataset (which does not appear to be available any longer at the link referenced in [43]) in addition to some that have been generated by themselves. This set is an improvement over the second Columbia set because of the representation of different subject categories and greater number of examples.

Figure 2.7: Example Authentic (Left) And Tampered (Right) Images From The CASIA TIDE Database V1

2.2.4 CASIA Tampered Image Detection Evaluation (TIDE) Database V2

Later in 2010, CASIA created the TIDE database v2 [1]. The CASIA TIDE database v2 is arguably the most extensive and challenging of these datasets, comprised of 7,200 authentic images and 5,193 tampered images. It is a superset of the CASIA TIDE database v1 but also brings new components and was made with the intention to provide more comprehensive tampering examples [43]. Like the CASA TIDE database v1, there are there are tampered region type classifications, and splicing operation classifications. The CASIA TIDE database v2 improves upon the CASIA TIDE database v1, however, by

introducing an additional subject category ("indoor"), and also tracks the size of the tampered region (small, medium, large) as well as incorporating post processing techniques onto the tampered image (blurring of the tampered region's boundary area or blurring in regions other than the boundary of the tampered region). The CASIA TIDE database v2 is also a multi-resolution dataset and gets its content again from the Corel database, some content from the CASIA TIDE database v2 creators, and also other websites that have provided permission. This dataset also features both compressed (JPEG) and uncompressed (BMP, TIF) content. As described by the creators of the CASIA TIDE datasets, "database v2.0 is more challenging and comprehensive compared with database v1.0 because of post-processing used, diversity of image formats and use of images from different sources" [43].

Figure 2.8: Example Authentic (Left) And Tampered (Right) Images From The CASIA TIDE Database V2

2.2.5 The Importance of Equalizing JPEG Compression Applications

In the current landscape of digital images, the most likely file format a user will encounter in their daily activities is a compressed JPEG. For this reason the CASIA TIDE

databases are the most alluring for research purposes in order to emulate the common use case. The CASIA TIDE database v2 is itself alluring because its inclusion of post processing techniques to create convincing forgeries and thus has the capability to test a passive detection framework most thoroughly. However, one publication identifies a need to be wary of blindly choosing the CASIA TIDE database v2 as the most rigorous – at least when using the database as-is.

[18] calls attention to the fact that there is a mismatch in the number of compression passes when comparing authentic images and tampered images in the CASIA TIDE database v2. Because many images in the authentic set are JPEG compressed in the first place, splicing a region from a JPEG-compressed image into another JPEG-compressed image and then compressing that new composite image again introduces a statistical bias.

The table from [18] below substantiates this claim by showing the mean, variance, skewness and kurtosis for the luminance and chrominance channels of the edge images two source images (J_1 and J_2). J_1 is created by taking a random uncompressed image (ucid00017.tif from [44]) and subjecting it to JPEG compression with quality factor = 95. This means that J_1 has chrominance subsampling (4:2:0). J_2 takes J_1 and compresses it with Adobe Photoshop with level 11. J_1 has the resolution of its chrominance channels reduced by a factor of 2 while J_2 does not.

Channel	Image	μ	σ^2	γ_1	γ_2
Y	J_1	5.99	131.67	3.19	16.99
	J_2	5.96	130.43	3.20	17.12
Cb	J_1	0.44	0.86	3.83	28.53
	J_2	0.60	2.19	5.80	62.19
Cr	J_1	0.39	0.67	3.12	18.10
	J_2	0.58	1.82	4.39	32.67

Table 2.1: Comparison Of Edge Image Statistics [18]

This statistical bias does indeed influence classification accuracy rates as will be shown in Section 3.3.1. However, the CASIA TIDE database v2 data can still be used for testing with this bias in mind and some preprocessing on the authentic and tampered sets which will be described later.

3 PROPOSED FRAMEWORK

With an understanding of techniques that are currently available this thesis investigates how to further the field of passive image splicing detection. Commonly a new research endeavor will propose a novel approach for extracting features. As can be seen in Section 2.1.2 this mindset has done much to uncover various perspectives from which to address this problem. However, there is not much attention paid to the concept of preprocessing the input images as a way to strengthen existing approaches. This study makes preprocessing a focus and works off of the Markovian rake feature extraction technique put forth by the authors in [18]. This reference study was chosen on the basis that its accuracy rates are competitive with other modern approaches, as will be shown in Section 3.3.1. Moreover, the publication describing the reference framework is unique in the attention that it pays to the issue of authentic and tampered datasets with unequal processing as was described in Section 2.2.5.

In the following parts of this chapter, the high-level overview of the framework will be discussed, and more detailed descriptions of each of the steps will follow.

3.1 FRAMEWORK OVERVIEW

As was discussed in Section 2.1.2, many passive image tampering detection frameworks follow the same general form. The process utilized in this project observes this general framework but also spends a considerable amount of time investigating the effectiveness of various preprocessing techniques. The high level framework followed in this project can be found below:

1. Selection of images for testing
2. Preprocessing of images
3. Feature extraction from test images
4. SVM classification

The following sections of this chapter will provide a more detailed description of each of the four steps listed above.

3.2 Selection of Images for Testing

Because the CASIA TIDE database v2 [1] is the most challenging dataset available, it was chosen as the dataset to test the proposed solution. In order to provide the same number of authentic and tampered images to the classifier in a later step, subsets of 5,000 authentic images and 5,000 tampered images were randomly chosen to conduct testing. These members were chosen by using the `randperm` [45] function in MATLAB to create a seeded random vector of indexes, based on the size of the authentic and tampered datasets. The first 5,000 of these indexes were then provided to the lists of authentic and tampered images to extract a random sampling from the entirety of their respective full sets. These randomly selected sets for authentic and tampered images were saved to provide a consistent basis for comparison between exploratory testing implementations.

3.3 Preprocessing of Images

Before subjecting each image to feature extraction processes there are a number of preprocessing techniques that were explored to examine their effect on accuracy rates. At

a high level, these techniques can be grouped into JPEG quality factor equalization, color space conversion, JPEG block line blurring, and image level filtering.

3.3.1 JPEG Quality Factor Equalization

Section 2.2.5 introduced the importance of equalizing the number of JPEG compression passes applied to tampered and authentic content. The claim that this mismatch in the number of JPEG compression iterations applied to authentic and tampered content is in fact substantiated in trial runs, even more so than in the statistical mismatches identified in *Table 2.1*. Using the feature extraction methodology prescribed in [18], we can see in *Table 3.1* that accuracy rates are outstanding when the CASIA TIDE database v2 data is used as-is. The results below show detection rates when only features from the Y channel are used, when features from the Y and Cb channels are used, and when features form all three channels are used.

Channels	TPR	TNR	AR
Y	86.99 %	91.37 %	**89.19** %
YCb	95.88 %	97.19 %	**96.54** %
YCbCr	95.12 %	97.59 %	**96.36** %

Table 3.1: Performance of Reference Algorithm on Un-Equalized Quality Factors

[18] proposes a process for preprocessing the CASIA TIDE database v2 authentic and tampered datasets which should eliminate the unfair statistical biases noted in the previous chapter:

1. Recompress 7,437 JPEG authentic images by MATLAB with quality factor = 84
2. Compress 3,059 TIFF tampered images by MATLAB with quality factor = 84

3. Leave 2,064 JPEG tampered images untouched.

Recompressing the JPEG-compressed authentic subset with a quality factor of 84 (using the '`quality`' flag in MATLAB's `imwrite` [46]) puts this subset of images at the same number of compression passes as the JPEG tampered content. This is because JPEG tampered content likely comes from two authentic JPEG images which have been JPEG compressed again once the splicing operation has occurred.

Compressing the TIFF tampered images addresses the case where two JPEG authentic images were combined into one via image via splicing techniques but the composite image is not compressed again. Subjecting these images to another pass of JPEG compression puts this content in the same position as the previous set subjected to preprocessing.

The remaining JPEG tampered images do not require any preprocessing at this stage because they foreseeably have been subjected to the same number of compression passes as the previous two sets.

We can see that this preprocessing does in fact make a difference and has a considerable impact on the detection rates:

Channels	TPR	TNR	AR
Y	76.11 %	80.40 %	**78.28 %**
YCb	79.42 %	85.17 %	**82.31 %**
YCbCr	81.98 %	84.09 %	**83.05 %**

Table 3.2: Performance Of Reference Algorithm On Dataset With Equalized Quality Factors

A difference of roughly 13.5% in accuracy rates between the optimal cases in each configuration draws indisputable attention to the necessity of this preprocessing. The significant drop-off in accuracy rates can be considered an indication of the "toughness" of the problem incurred by this JPEG quality factor equalization preprocessing.

It is the belief of this author that the a framework's effectiveness for solving the image splicing tampering detection problem should be done with respect to the worst case conditions. This means that the authentic and tampered images should be evaluated without any known biases influencing results. By eliminating this known quality factor mismatch bias, a framework can be judged by the quality of the differentiating features it extracts. For this reason, the compression application equalizing technique described previously is used to preprocess the CASIA TIDE v2 authentic and tampered images for use in this project.

More recent image splicing approaches such as [20] and [21] use the CASIA TIDE v2 dataset but do not account for the quality factor mismatch and have accuracy rates of 98% and 93% respectively. Because [18] has performance that is in line with these approaches on the same dataset when this quality factor equalization preprocessing is not done, it is fair to say that the reference approach is competitive with modern approaches. Moreover improving on the reference approach means that this research contributes to a modern use case and is not just improving a weaker implementation.

3.3.2 Color Space Conversion

Due to the data structure of a color image, processing must always be done on three separate layers. Image data is often represented in three color planes with each containing

intensity values for red, green, and blue. This is commonly referred to as the RGB color space. It is the way that digital images are first captured using color filter arrays (CFAs) which use different filtered sensors that capture red, green, and blue light separately much like different cones in the human eye. Interpolation methods (called demosaicing [47]) then construct full planes for red, green, and blue information which are overlaid to produce a color image with an additive color model.

Although the RGB color space is powerful in that it is extremely intuitive, there is also value in transforming the RGB information to keep its luminance (light intensity information) and chrominance (color information) information separate. A number of color space conversions achieve this but the conversion from RGB to YCbCr is one of the most popular. To represent image data in the YCbCr color space, the following conversion equations can be applied to an image pixel that is in the RGB color space:

$$Y = (0.299 \times R) + (0.587 \times G) + (0.114 \times B) \qquad \textit{Equation 1}$$

$$C_b = -(0.299 \times R) - (0.587 \times G) + (0.886 \times B) \qquad \textit{Equation 2}$$

$$C_r = (0.701 \times R) - (0.587 \times G) - (0.114 \times B) \qquad \textit{Equation 3}$$

Luminance information is contained in the Y component and is a weighted sum of the red, green, and blue values for a pixel while the chrominance information is stored with respect to blue and red in the Cb and Cr channels. Although MATLAB makes this conversion simple to execute at a high level using the function `rgb2ycbcr` [48], documentation acknowledges observation of the International Telecommunication Union standard which specifies the coefficients listed above in the transition matrix [49].

Converting to the YCbCr color space is useful because while it is now known that the human vision system is much more sensitive to changes in luminance information than chrominance information, the properties of chrominance channels can sometimes help identify evidence of splicing tampers. The implication of this observation is that tampering attempts that may be imperceptible to the naked eye can reveal itself in the chrominance channels.

The framework of [18] proposes the omission of chrominance channels in the image splicing tampering detection framework. As can be seen in *Table 3.1* and *Table 3.2*, however, there is value in including features extracted from these planes into an image's feature vector. There is an increase in accuracy rates of at least 4% that is seen in all implementations. [50] acknowledges that "[s]ince human are more sensitive to luminance than to chrominance, even if spliced image looks natural to human, some unnatural clues will be left in chrominance channel". Indeed multiple publications written by Wang, Dong and Tan ([50], [22]) emphasize the effectiveness of including chrominance channels to image splicing tampering detection frameworks with a marked increase in accuracy.

3.3.3 JPEG Block Line Blurring

The reference approach uses DCT to represent the frequency domain content of the images it processes. This is because splicing, from a human perspective, is often identified due to unnatural edges surrounding the spliced region. If a feature vector can facilitate the identification of unusual frequencies and edges associated with tampered regions then it is successful. For this reason, it is desirable to eliminate any unnatural frequencies or edges that exist in images that are regularly occurring.

Common to all JPEG processing, all channels (YCbCr) are subjected to 8×8 non-overlapping block DCT as can be seen in *Figure 3.1*. Due to coarse quantization of low-frequency DCT coefficients, a mosaic pattern may appear in the decompressed image even in smooth regions [51]. This can also lead to a regular pattern observable in JPEG-compressed images, which is visibly unnatural and potentially can produce frequencies or edges that can interfere with the detection of those associated with spliced regions.

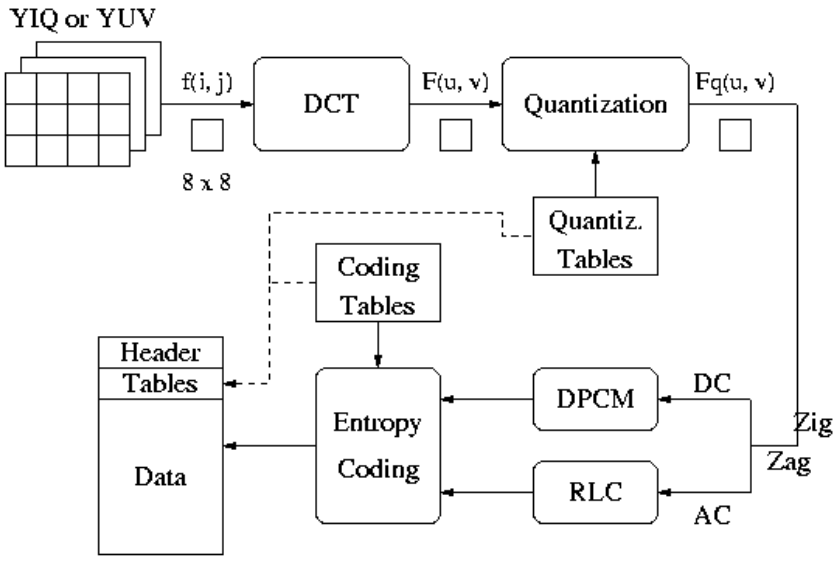

Figure 3.1: JPEG Processing Block Diagram [52]

Figure 3.2 below shows an image with mostly smooth regions while *Figure 3.3* zooms in on a region of *Figure 3.2* a region where block-based processing is particularly evident in a regular pattern. Because the reference image has been zoomed in so much, it is possible to differentiate between each pixel. Counting the number of pixels between regularly appearing pattern artifacts, affirms that these patterns coincide with the 8×8 blocks used by the JPEG algorithm.

Figure 3.2: JPEG-Compressed Image With Evident Block Lines (Au_Nat_00093.jpg)

Figure 3.3: Zoomed-In Region Of JPEG-Compressed Image With Evident Block Lines (Au_Nat_00093.jpg)

Figure 3.4 below shows the same image as in *Figure 3.2* after the JPEG block lines have been blurred. *Figure 3.5* zooms in on the same region captured in *Figure 3.3* to provide to show the visual effect of blurring along JPEG block lines.

Figure 3.4: Au_Nat_00093.Jpg After Block Line Blurring

Figure 3.5: Zoomed-In Region Of Au_Nat_00093.jpg After Block Line Blurring

It is apparent in comparing the zoomed in regions of the reference image before and after block line blurring that this regular pattern becomes somewhat less pronounced. This should reduce any effect this regular pattern has on distracting the classifier and draw more attention to the "real" unnatural edges produced by splicing tampers.

Figure 3.6 shows a sample 8×8 block in a given image with the borders to be blurred shown in red. The numbers in the border pixels and adjacent pixels demonstrate an example weighting scheme for a weighted average of boundary pixels.

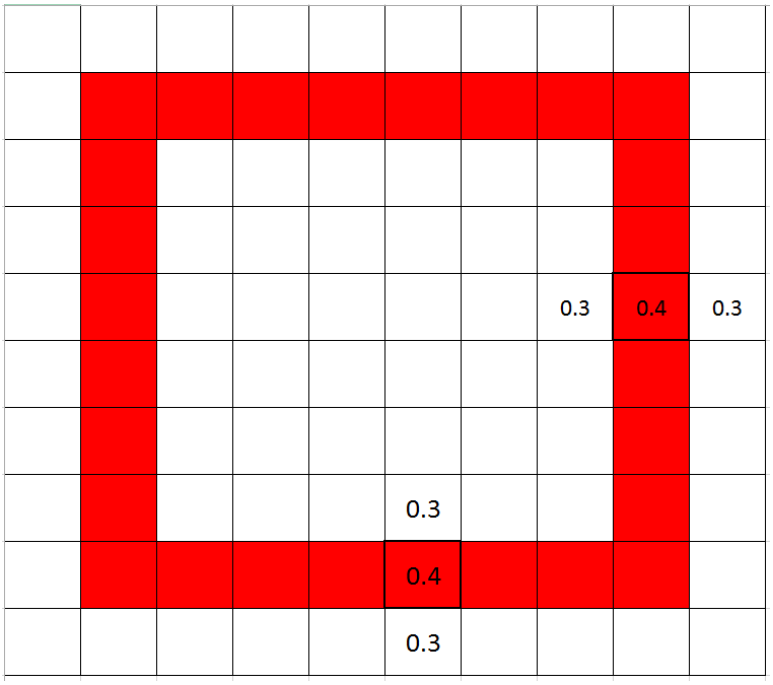

Figure 3.6: Illustration Of Block Line Blurring With Sample Coefficients

Figure 3.7 visualizes an image level DCT of Au_nat_00093.jpg without any JPEG block line blurring applied along with three different weighting schemes to show the effect of the proposed blurring on the frequency content of the image.

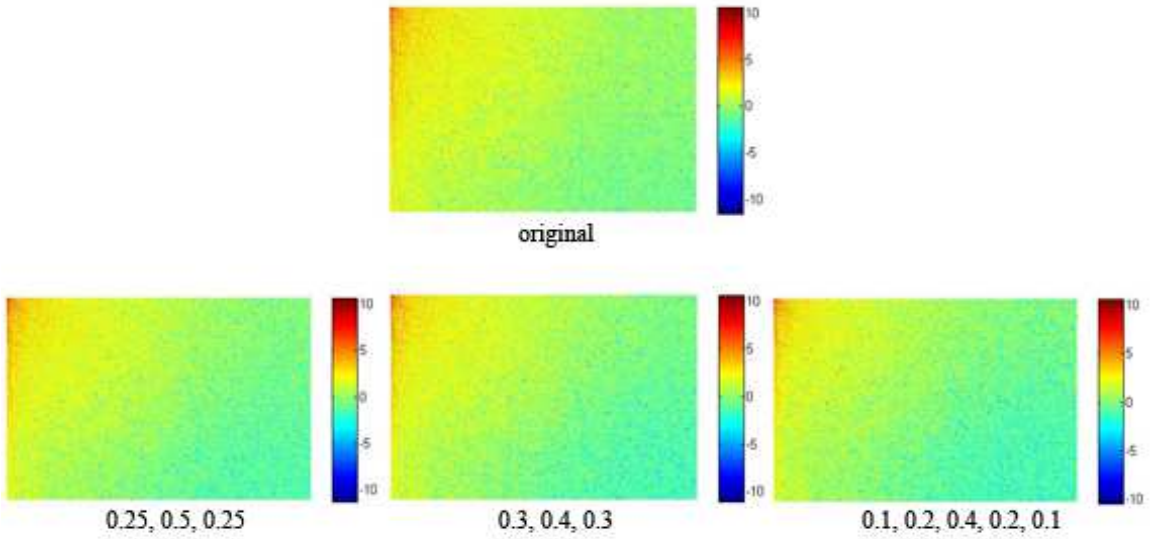

Figure 3.7: Impact Of Changing JPEG Block Line Blurring Weightings On Frequency Content

The above figure shows that the proposed block line blurring scheme does affect the frequency content of the images but only slightly which is to be expected. The lower right hand corner (higher frequencies) is expressing a slightly cooler coloration than in the original, suggesting that the block line blurring was effective in removing the higher frequencies associated with the regularly occurring block patterns.

3.3.4 Image Level Filtering

With similar goals of blurring block line boundaries to reduce the impact of JPEG-induced unnatural edges, another preprocessing technique explores the effectiveness of globally processing an image. Due to the focus on edges in this analysis, the two high level processing techniques explored were blurring and sharpening.

3.3.4.1 Image Level Blurring

In the same way that JPEG block boundary blurring aims to remove high frequencies induced by block processing, image level blurring also has the intention to remove the highest frequency noise in an image. To blur an image, MATLAB provides a number of parameters that can be tuned.

Blurring is a two-step process using MATLAB, beginning with the creation of a filter using `fpsecial` [53] that is then applied to an image using `imfilter` [54]. Customization of the filter is mostly available within `fspecial(type, parameters)`. Details for averaging and Gaussian blurring filters are below:

Type	Description	Parameters
average	(averaging filter)	• hsize: size of the filter (3x3 by default)
Gaussian	(Gaussian lowpass filter)	• hsize: size of the filter (3x3 by default) • sigma: standard deviation (0.5 by default)

Table 3.3: MATLAB Blurring Filter Types and Parameters

The idea behind image level blurring as a preprocessing technique in this study is to essentially eliminate higher frequency noise. This is to say that the degree of blurring is ideally not high. An example of the slight amount of blurring targeted for this preprocessing stage is shown in *Figure 3.8*.

Figure 3.8: Au_ani_10208.jpg Original (left) And Gaussian Blurred With $\sigma = 0.5$ *And* $hsize = 3x3$ *(right)*

As can be seen by *Figure 3.8*, blurring an image with such low intensity does little to visually change the image but it does have a perceptible impact on the frequency content of the image. *Figure 3.9* visualizes an image level DCT of Au_ani_10208.jpg without any blurring applied. The lower right hand corner of this figure is where the high frequencies in the image are represented and the "warmth" of its coloration indicates the amount of information there. *Figure 3.10* provides the same visualization for Au_ani_10208.jpg after Gaussian blurring has been applied with filter size 3×3 and $\sigma = 0.5$. Comparatively, the lower right hand region is represented by much cooler colors, indicating some smoothing and reduction in higher frequencies, as is desired.

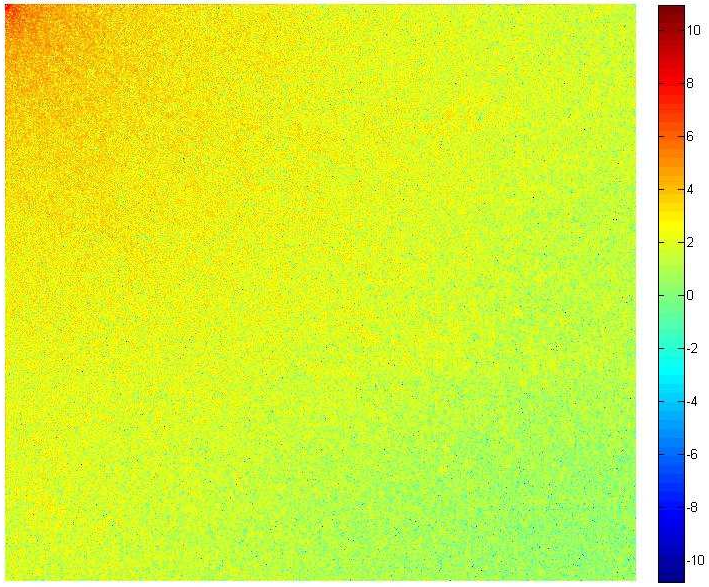

Figure 3.9: Frequency Content Of Au_Ani_10208.jpg (Original)

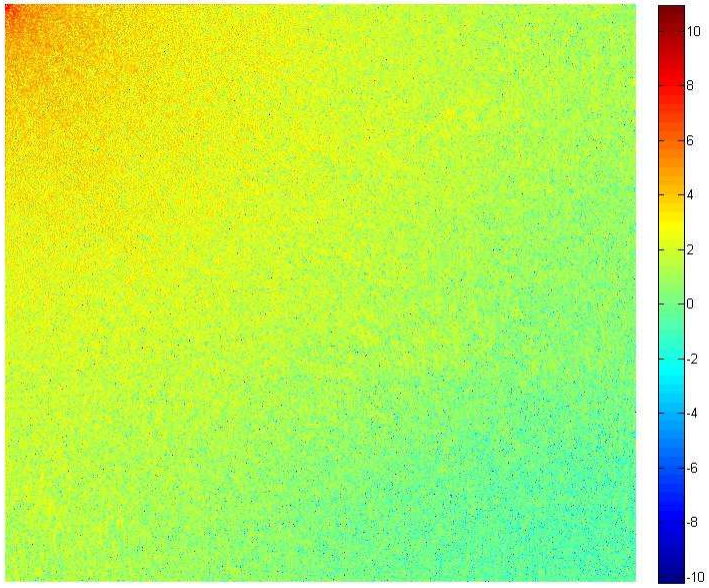

Figure 3.10: Frequency Content Of Au_Ani_10208.jpg After Gaussian Blur

To understand the effect of modifying the filter size and σ parameters for Gaussian blur filter generation, the figure below shows how the frequency information responds to these changes. Note that the notation "$hsize = W \times H$" will be used in the rest of this thesis to indicate a filter size that equals the size of the image it will be blurring.

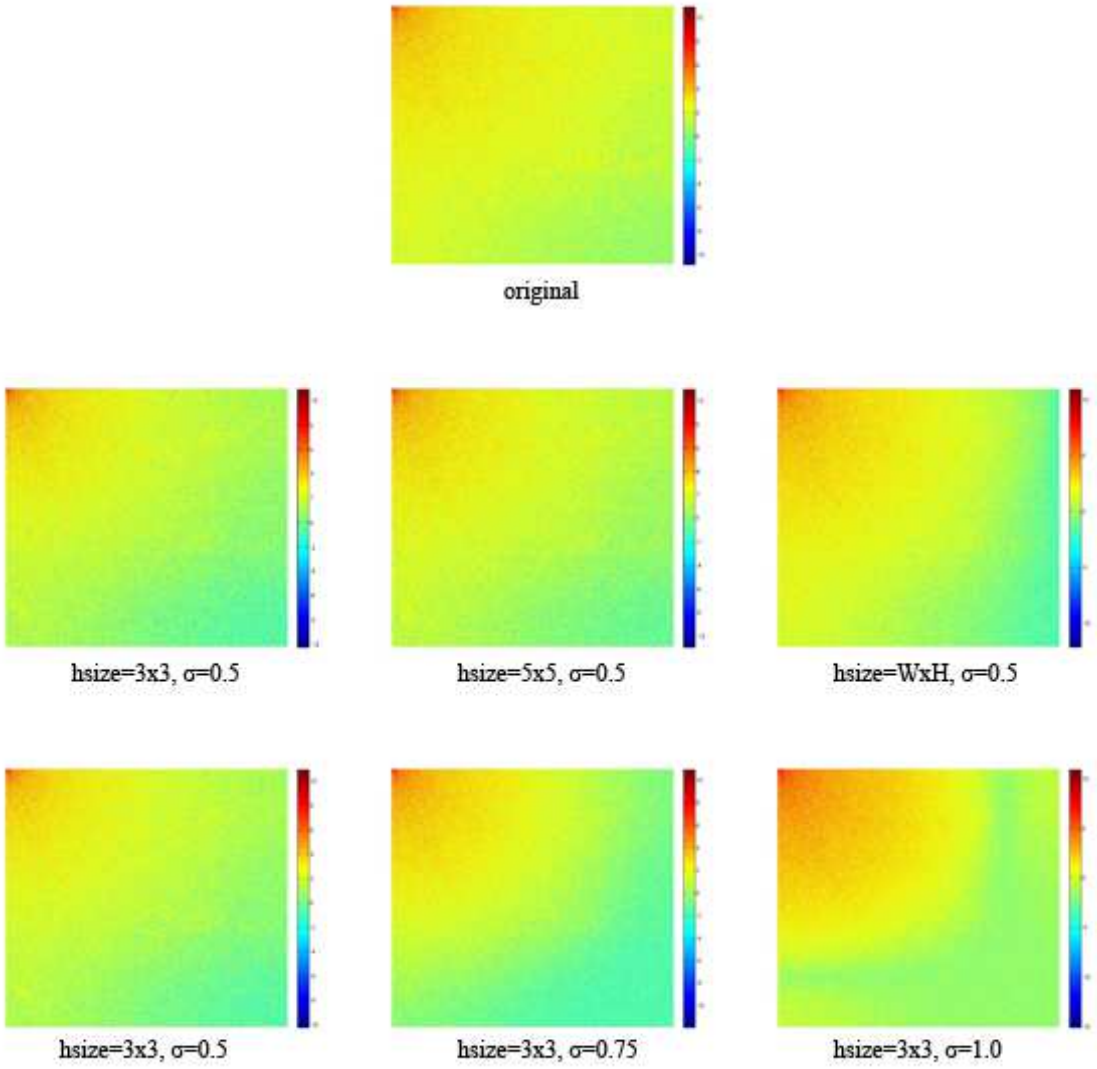

Figure 3.11: Impact of Changing Gaussian Blurring Filter Parameters On Frequency Content

Figure 3.11 illustrates that for Gaussian blurring, both the filter size and σ impact what will later be called the "blur amount" which is tied to the degree with which the frequency information is modified.

To understand how the blurring with the averaging filter affects an image's frequency information, the figure below shows the DCT representation for three different filter sizes:

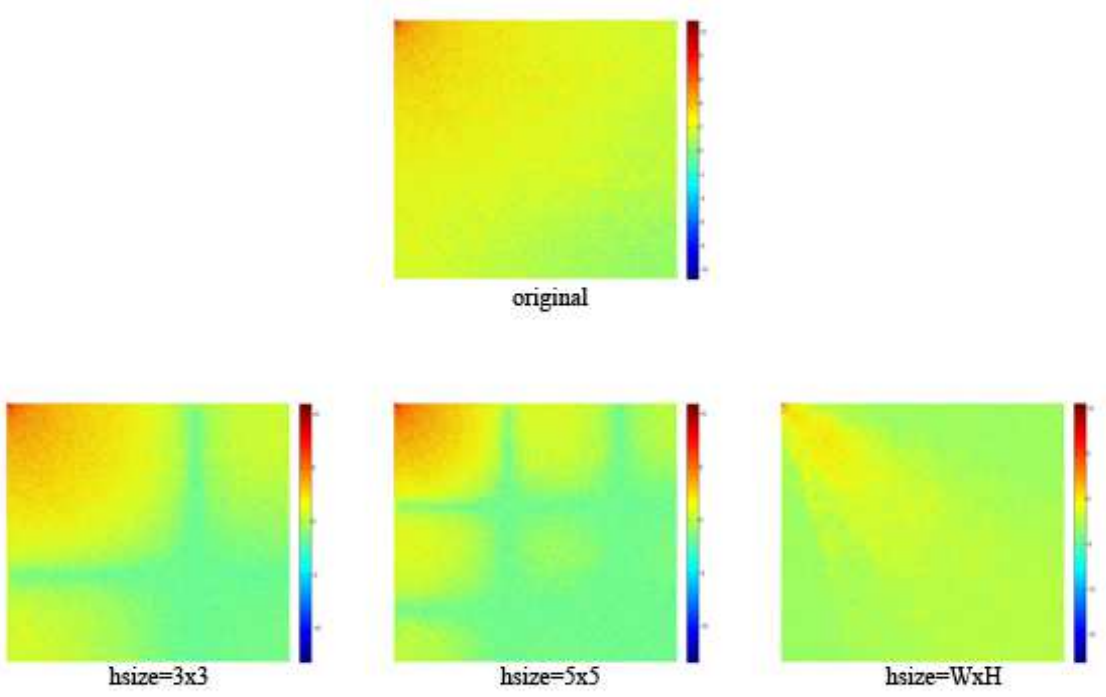

Figure 3.12: Impact of Changing Averaging Blurring Filter Parameters On Frequency Content

Figure 3.12 shows that changing the filter size when blurring with the averaging filter has a large impact on the frequency information in the image. Selecting $hsize = W \times H$ changes the frequency content drastically, but also the visual content. A

comparison of visual impacts on resulting blurred images will be addressed in Section 4.1.3.1.1.

3.3.4.2 Image Level Sharpening

The motivation for sharpening an image in the preprocessing stage is to accentuate edges in the hopes that doing so will also highlight the differentiating features for spliced regions in tampered content. MATLAB has a built-in image sharpening feature called `imsharpen` [55]. This function has a number of parameters which can be tuned to achieve an optimal configuration.

Parameters for this function include the following:

Type	Description	Parameters
imsharpen	(sharpening filter)	• 'Radius': Standard deviation of the Gaussian lowpass filter (1 by default) • 'Amount': Strength of the sharpening effect (0.8 by default) • 'Threshold': Minimum contrast required for a pixel to be considered an edge pixel (0 by default)

Table 3.4: MATLAB Image Sharpening Filter and Parameters

As was indicated in Section 3.3.4.1 the goal for these preprocessing techniques is to modify the input images only slightly. For this reason, only a small amount of sharpening will be applied. An example comparison of an original and sharpened image is seen in *Figure 3.13*.

Figure 3.13: Au_Ani_10208.jpg Original (Left) And Sharpened With Amount = 0.75 (Right)

The figure below examines the effects of small amounts of sharpening on the frequency information of Au_ani_10208.jpg:

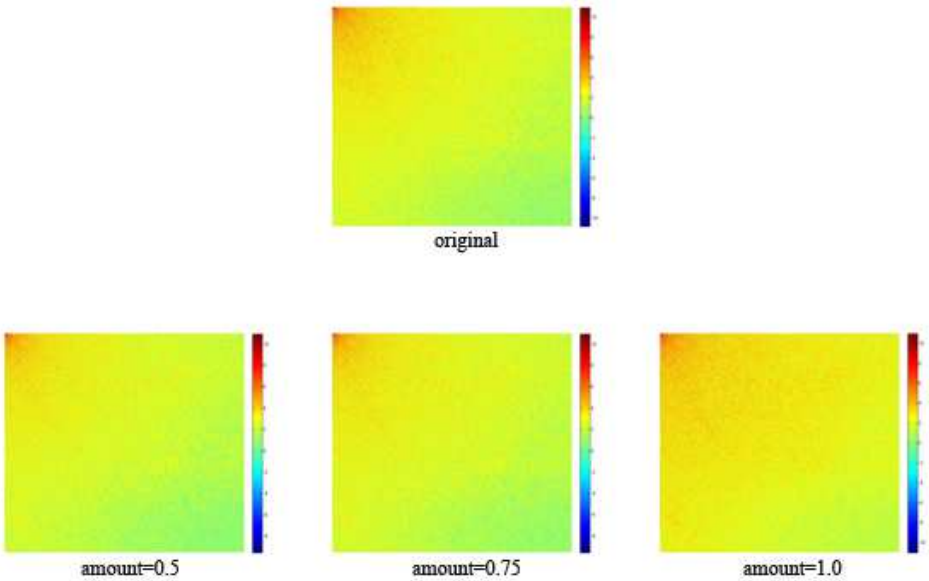

Figure 3.14: Impact Of Changing Sharpening Filter Parameters On Frequency Content

It can be observed in *Figure 3.14* the "warmth" of the colors between the upper left hand corner of the visualizations (lowest frequencies) and the lower right hand corner (highest frequencies) increases as the parameter '`amount`' increases.

3.3.5 Content Adaptive Techniques

While it is intuitive to apply preprocessing techniques uniformly across an entire set of authentic and tampered images, digital images are extremely diverse and an optimal processing configuration for one image may not be optimal for another. For this reason, it is worth investigating configurations that are determined by an image's content. Choosing the blur or sharpening amount, for instance, can be informed by the amount of higher frequency content present in an image. One way to quantify this is by using the two-dimensional DCT to recompose an image plane in the frequency domain. This will create a matrix with the same dimensions as the input image plane but represent the various 2D frequency components like the figure below illustrates:

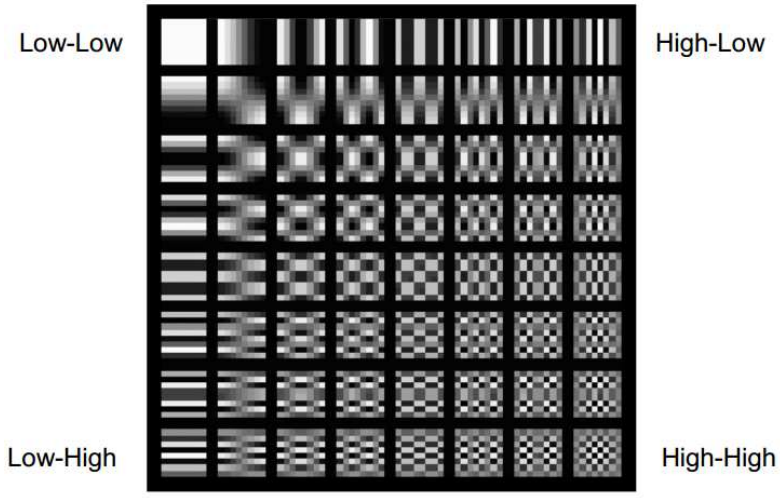

Figure 3.15: Illustration Of DCT Frequency Ordering [56]

The 2D DCT stores the average energy of the cell (low frequency) in the upper-left-most cell of the matrix and horizontal and vertical frequency increases moving right and down. As such the combined frequency in both the vertical and horizontal directions can be thought to increase in a zigzag pattern as shown below:

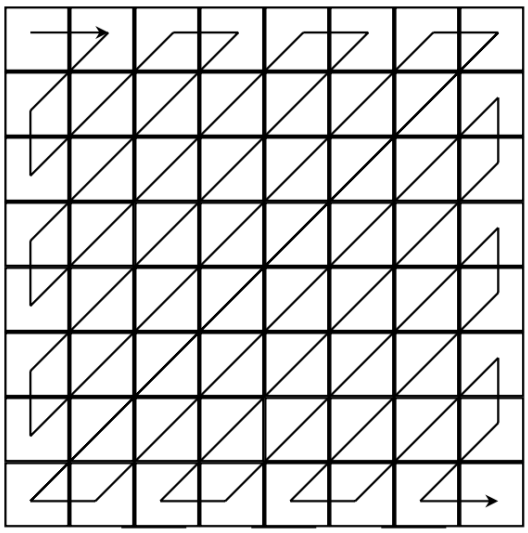

Figure 3.16: Zigzag Pattern Followed By DCT Coefficients [56]

With *Figure 3.16* serving as a reference point for extracting coefficients representing increasing combined frequency, this matrix can be transformed into a vector that sorts by combined frequency. Taking off the first 25% and the last 25% of coefficients in this vector can then provide coefficients for mid/high range frequencies. The average of this reduced vector (referred to as μ in this study) can serve as an approximation of the amount of mid/high frequency content that can serve as a basis of comparison between two images. This knowledge guides the process for classifying images with varying amounts of high frequency content.

From here, it's necessary to make meaningful assertions based on information gleaned from the above process. The first obvious question is what values of μ constitute an image with a particularly large amount of high frequency content. Figures below show scatterplot representations for μ of authentic and tampered content from the CASIA TIDE database v2 with quality factor equalization described in Section 3.3.1.

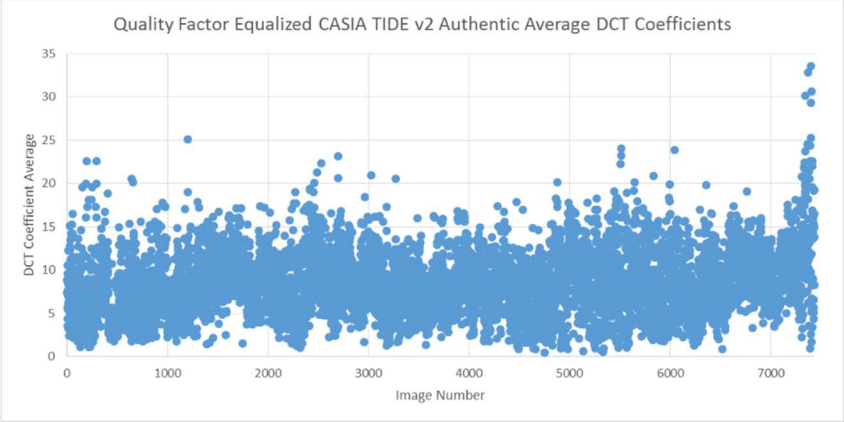

Figure 3.17: Scatterplot Of Quality Factor Equalized CASIA TIDE V2 Authentic Average DCT Coefficients (μ)

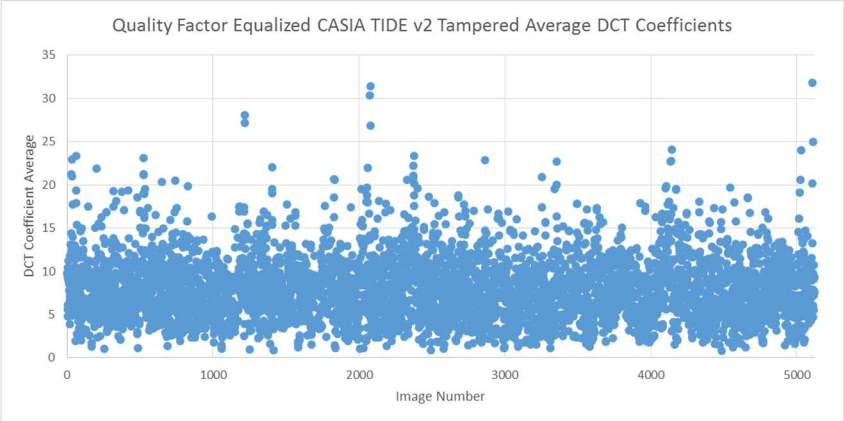

Figure 3.18: Scatterplot Of Quality Factor Equalized CASIA TIDE V2 Tampered Average DCT Coefficients (μ)

Using this factor μ as a basis for determining the amount of high frequency content can be checked by cross referencing an image with its μ. For the CASIA TIDE database v2 authentic set, the images with the highest and lowest μ are shown below:

Figure 3.19: Image With Lowest μ (Left, $\mu = 0.458$) And Highest μ (Right, $\mu = 33.57$) From The CASIA TIDE Database V2 Authentic Set

Visual inspection confirms that this value μ can be used as an effective indication of the degree of mid/high frequency content in a given image. The image with the lowest μ is in fact very smooth and does not have much detail. The image with the highest μ however is very detailed and contains an abundance of high frequency content. With validation that μ can be used as an indicator of the degree of higher frequency content in an image, Section 3.3.5.1 and Section 3.3.5.2 detail two approaches that use this value to determine adaptive configurations.

These adaptive configurations look to modify the "filter amount". As was detailed in *Table 3.3* and *Table 3.4* there are a number of parameters which can be adjusted to

achieve the maximum performance. These adjustments will be made as a result from results in Section 4.1.3.1 and Section 4.1.3.2.

3.3.5.1 Binning based on thresholds

The first adaptive approach is also the simplest. It uses the threshold μ value and chooses one filter amount if a given image has a μ greater than or equal to the reference and another blur amount if it has less than the reference:

$$filterAmount(\mu) = \begin{cases} val_1, & \mu < threshold \\ val_2, & \mu \geq threshold \end{cases} \qquad \textit{Equation 4}$$

For the authentic and tampered sets, the μ value for each individual image is seen in *Figure 3.17* and *Figure 3.18*. These arrays for each set can be called $L_{\mu_{auth}}$ and $L_{\mu_{tamp}}$. Let μ_N be the value for μ such that N percent of values in $L_{\mu_{auth}}$ or $L_{\mu_{tamp}}$ are less than μ_N. Changing the value for N can then determine different distributions for which val_1 and val_2 are assigned and μ_N can be used as the threshold value. The `prctile` [57] command in MATLAB makes this calculation easy to determine the various values for μ_N to be used as a threshold.

3.3.5.2 Determining filtering amount based on a linear function

While processing a given image based on two discrete bins may be sufficient, it is also possible that the unevenness of processing applied across the datasets will make classification tougher. To mitigate this possibility, a more continuous approach was devised for determining the filtering amount. Instead of binning images based on a threshold, the blurring/sharpening amount can be decided using a simple linear function:

$$filterAmount(\mu) = base + modifier * (\frac{\mu}{\mu_{\max}}) \qquad \textit{Equation 5}$$

μ_{max} is the maximum value across the entire tampered and authentic dataset. After this maximum value is determined, each image can be individually evaluated and processed with respect to that. This μ_{max} can be determined for all three channels (YCbCr) and used to blur each channel adaptively. Between the authentic and tampered sets, μ_{max_y} was found to be 33.6, $\mu_{max_{cb}}$ was found to be 5.86 and $\mu_{max_{cr}}$ was found to be 5.28 when examining the datasets after quality factor equalization.

It is important to realize that this value must be the maximum of both datasets combined because taking the respective μ per each dataset and using this value as a modifier can unfairly bias feature extraction for the authentic and tampered sets. This is because the two datasets are effectively being processed differently and can lead to statistical indicators which the classifier can leverage.

3.4 FEATURE EXTRACTION

As has been previously addressed, the method for feature extraction is the same as was proposed in [18]. The pseudocode for feature extraction is shown below:

1. Translate the image into the YCbCr color space. Start with the luminance channel
    a. Process the image with $n \times n$ block DCT ($n = 4$ first), producing another matrix of size $W \times H$. Call it $F_{n \times n}$

i. Calculate difference array in the horizontal direction, round results, threshold values above 6 and below -6. Call it D_h

$$D_h(i,j) = F_{n\times n}(i,j) - F_{n\times n}(i+1,j) \qquad \textit{Equation 6}$$

ii. Calculate difference array in the vertical direction, round results, threshold values above 6 and below -6. Call it D_v

$$D_v(i,j) = F_{n\times n}(i,j) - F_{n\times n}(i,j+1) \qquad \textit{Equation 7}$$

iii. Create a transition probability matrix for D_h. Call it $TPM_{h\,|\,n\times n}$

$$p\{D_h(u+1,v) = i \mid D_h(u,v) = j\} = \frac{\Sigma_{u,v}\delta(D_h(u,v) = j, D_h(u+1,v) = i)}{\Sigma_{u,v}\delta(D_h(u,v) = j)} \qquad \textit{Equation 8}$$

iv. Create a transition probability matrix for D_v. Call it $TPM_{v\,|\,n\times n}$

$$p\{D_v(u+1,v) = i \mid D_v(u,v) = j\} = \frac{\Sigma_{u,v}\delta(D_v(u,v) = j, D_v(u,v+1) = i)}{\Sigma_{u,v}\delta(D_v(u,v) = j)} \qquad \textit{Equation 9}$$

v. Save $TPM_{h\,|\,n\times n}$ and $TPM_{v\,|\,n\times n}$

vi. Repeat a. with $n = 8$ and then $n = 16$

b. Concatenate $TPM_{h\,|\,4\times4}$, $TPM_{h\,|\,8\times8}$, $TPM_{h\,|\,16\times16}$, $TPM_{v\,|\,4\times4}$, $TPM_{v\,|\,8\times8}$, $TPM_{v\,|\,16\times16}$ into one feature vector for that

channel. Call this $TPM_{all\,|\,p}$ where p indicates the plane represented (Y, Cb, Cr)

c. Repeat 1. with the chrominance channel Cb and then the chrominance channel Cr

2. Concatenate $TPM_{all\,|\,Y}, TPM_{all\,|\,Cb}, TPM_{all\,|\,Cr}$. Call this $TPM_{all\ planes}$

Although the methodology for feature extraction is taken from [18] it is important to note that unlike their approach, feature extraction is expanded to include the chrominance channels in addition to the luminance channel. The reasoning for this has been explained further in Section 3.3.2.

3.5 CLASSIFICATION

In this project, a support vector machine (SVM) is used to perform classification of authentic and tampered images. For SVMs, classification starts with training the classifier by providing it with a collection of feature vectors along with a label vector indicating which data class it belongs to (authentic or tampered in this case). This collection of vectors is called the training set. In instances where there are only two data classes (such as in this study), classification using the training set can be visualized like in *Figure 3.20*. It should be noted that this figure represents feature vectors with two dimensions but the principle behind it can be expanded to the n^{th} dimension.

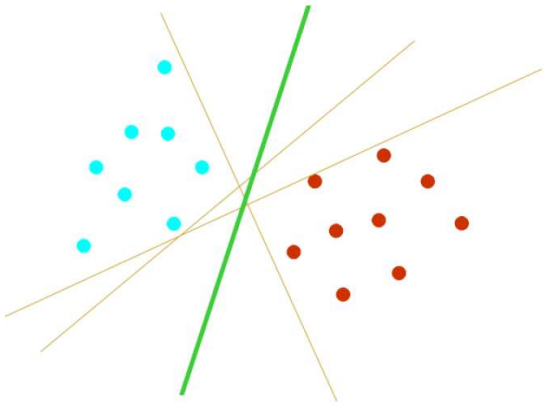

Figure 3.20: 2D Projection Of Features With Two Classes [58]

Although this is a simple example (the features are only two-dimensional) it illustrates the basic principle in two-class classification. The green line represents the hyperplane which aims to optimally separate the two sets of training vectors (light blue dots and dark red dots). The hyperplane is considered optimal based on two conditions:

1. The partition established by the hyperplane keeps the two classes completely distinct (i.e. no errors)
2. The hyperplane is separated from the closest vector by the greatest possible amount

Item 2 is important because there can be multiple possible hyperplanes that can be chosen to "solve" the classification problem. Maximizing the distance between the hyperplane and the nearest vector aims to provide tolerance for future items to test that were not indicated in the training set. The light brown lines in *Figure 3.20* are possible hyperplanes that do not optimize distance from the nearest vector. Of course, satisfying

Item 1 (a property referred to as “linear separability”) is not always possible. An example of this is shown in *Figure 3.21*.

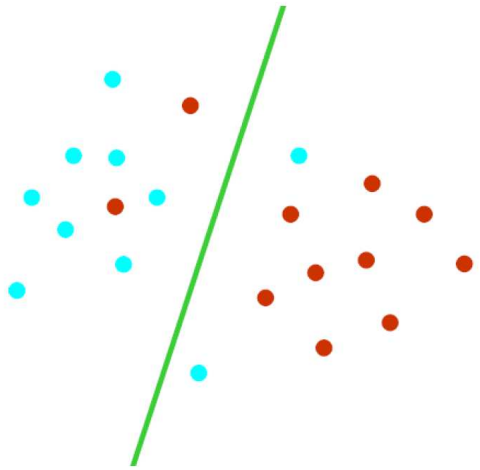

Figure 3.21: Classes That Are Not Linearly Separable [58]

In cases where classes are not linearly separable, a cost function [58] can help to keep a hyperplane accountable for classification errors. In this scenario, the goal is now to minimize this cost while also minimizing the capacity of a set of functions (also called the Vapnik–Chervonenkis dimension). [59] describes the VC dimension as “a way of measuring the complexity of a class of functions by assessing how wiggly its members can be”. Whereas high-degree polynomials lend to a more “wiggly” shape, minimizing the capacity of a set of functions translates to a less complex hyperplane shape.

In this project libSVM [12] was used within MATLAB to serve as a layer of abstraction to facilitate the complex process of training and testing the SVM classifier. With libSVM, the training step is handled with the function `svmtrain` which takes in the label vector and the feature vector and outputs a model. The testing step is handled with

the function `svmpredict` which takes in the model from the previous step and as well as the label vector and the feature vector and outputs a predicted label vector. This predicted label vector can be checked against the input label vector to generate a confusion matrix (`confusionmat` in MATLAB [60]) which give the values below. It should be kept in mind, however, that the mapping between positive/negative and authentic/tampered is subjective.

- True Negatives (TN): Number of tampered images properly classified as tampered
- True Positives (TP): Number of authentic images properly classified as authentic
- False Positives (FP): Number of tampered images improperly classified as authentic
- False Negatives (FN): Number of authentic images improperly classified as tampered

These values can then provide the following percentages:

- $True\ Negative\ Rate = \frac{TN}{TN+FP} \times 100$ *Equation 10*
- $True\ Positive\ Rate = \frac{TP}{TP+FN} \times 100$ *Equation 11*
- $False\ Positive\ Rate = \frac{FP}{TN+FP} \times 100$ *Equation 12*
- $False\ Negative\ Rate = \frac{FN}{TP+FN} \times 100$ *Equation 13*

- $Accurary\ Rate = \frac{TN+TP}{TN+TP+FP+FN} \times 100$ *Equation 14*

As was indicated in a Section 3.2, a randomly-chosen subset of 5,000 authentic and 5,000 tampered images were selected from CASIA TIDE v2. By using 10-fold cross validation, an average Accuracy Rate (AR) can evaluate the effectiveness of the features that have been extracted. 10-fold cross validation works as follows:

1. Randomly select 1/10 of the authentic image feature vectors and 1/10 of the tampered image feature vectors to test the classifier
2. Use the remaining 9/10 of the authentic image feature vectors and 9/10 of the tampered image feature vectors to train the classifier
3. Repeat this for a total of 10 times, choosing a different testing set and training set every time.

In this methodology the MATLAB function `crossvalind` facilitates the selection of the training and test sets. As specified by the MATLAB documentation, when designating the '`Kfold`' flag, `crossvalind` "returns randomly generated indices for a K-fold cross-validation of N observations. Indices contains equal (or approximately equal) proportions of the integers 1 through K that define a partition of the N observations into K disjoint subsets. […] In K-fold cross-validation, K-1 folds are used for training and the last fold is used for evaluation. This process is repeated K times, leaving one different fold for evaluation each time" [61].

4 ALGORITHM IMPROVEMENTS AND QUALITY FACTOR MISMATCH EFFECT

MATLAB 2013b was used to develop and test the methodologies that have been outlined in previous sections. Section 3.2 specified that a randomly selected consistent subset of the CASIA TIDE database v2 is used as a basis for comparison between different potential enhancements. This subset consists of 5,000 authentic images and 5,000 tampered images that represent each of the different image categories. As mentioned in Section 3.5, libSVM was used within MATLAB to facilitate the classification stage of this project. 10-fold cross validation is used to provide an average true positive rate (TPR), average true negative rate (TNR) and average accuracy rate (AR). Definitions for these terms can also be found in Section 3.5. "Positive" and "negative" can be assigned arbitrarily, and in this thesis they are used to represent authentic images and tampered images respectively. This is to say that the true negative rate is the rate at which the classifier accurately identifies tampered images as tampered, et cetera. Statistics for TPR, TNR, and AR are reported for feature extraction from all three image channels (YCbCr) due to the observation that, on average, using all three provides the highest rates.

Although this thesis uses the framework proposed by [18] Section 3.3 indicates the areas in which improvements were attempted and sometimes achieved. Section 4.1 details results of testing methodologies described in Section 3.3 while Section 4.3 examines the effect of varying JPEG quality factors between authentic and tampered datasets.

4.1 EXPERIMENTAL RESULTS FOR PREPROCESSING TECHNIQUES

This section explores the preprocessing techniques detailed in Section 3.3. The effect of choosing a color space for processing, blurring along JPEG block line boundaries, filtering an entire image, and combining these techniques are detailed in the sections to follow.

4.1.1 Experimental Results for Color Space Conversion

Table 4.1 examines the difference between running the reference algorithm [18] in the RGB color space as opposed to the YCbCr color space.

Channels	TPR	TNR	AR
RGB	76.43 %	82.34 %	**79.40 %**
YCbCr	81.98 %	84.09 %	**83.05 %**

Table 4.1: Reference Algorithm Performed In RGB And YCbCr Color Space

4.1.1.1 Effect of Selecting the Optimal Color Space

Although Section 3.3.2 provides reasoning for conducting testing in the YCbCr color space, *Table 4.1* proves through comparison that it outperforms the RGB color space when tested on the same splicing detection framework. For this reason, feature extraction is taken from the YCbCr representation of images in the rest of the results.

4.1.2 Experimental Results for JPEG Block Line Blurring

Three different configurations were tested in examining the effect of blurring along block lines induced by JPEG processing. The first two use only one neighbor pixel neighboring both sides of the pixel of interest, assigning different weights to the neighbors

and pixel of interest. The final configuration uses two neighbor pixels of interest. *Table 4.2* details the results of this testing:

Blur Weights	TPR	TNR	AR
0.25, 0.5, 0.25	83.14 %	85.19 %	**84.18 %**
0.3, 0.4, 0.3	82.76 %	85.21 %	**84.00 %**
0.1, 0.2, 0.4, 0.2, 0.1	83.15 %	85.49 %	**84.33 %**

Table 4.2: Results For JPEG Block Blur With Various Weights And Neighborhoods

4.1.2.1 Effect of JPEG Block Line Blurring With Various Weights and Neighborhoods

From these results it is clear that there is value in blurring around JPEG block boundaries, gaining up to 1.28% in accuracy over the reference approach. The greatest bump in accuracy was achieved using the 0.1, 0.2, 0.4, 0.2, 0.1 weighting scheme although the other two weighting schemes still outperformed the base reference by at least 0.95%.

4.1.3 Experimental Results for Image Level Filtering

Image level filtering examines the two possibilities that either the highest frequencies in an image should be lowpassed and reduced (slight amounts of blur) or higher frequency content should be augmented (slight amounts of image sharpening). The following sections provide the results for these two filtering approaches.

4.1.3.1 Experimental Results for Image Level Blurring

Image level blurring explores the possibility that feature extraction benefits from first lowpassing the image slightly to remove the highest frequencies. The default

averaging lowpass filter has a filter size (hsize) 3×3, while the default Gaussian lowpass filter has a filter size (hsize) 3×3 and $\sigma = 0.5$.

The table below examines the effect of changing the size of the averaging filter on the accuracy rate:

Parameters	TPR	TNR	AR
$hsize = 3 \times 3$ (default)	82.02 %	85.92 %	**83.98 %**
$hsize = 5 \times 5$	81.18 %	85.47 %	**83.34 %**
$hsize = W \times H$	62.6 %	61.0 %	**61.8 %**

Table 4.3: Results For Modifying Size Of Averaging Filter

Gaussian lowpass filters have two parameters which can be modified to achieve optimal performance. The table below shows the results of first modifying the filter size and then σ:

Parameters	TPR	TNR	AR
$hsize = 3 \times 3$ $\sigma = 0.5$ (default)	81.79 %	85.31 %	**83.56 %**
$hsize = 5 \times 5$ $\sigma = 0.5$	81.94 %	85.25 %	**83.61 %**
$hsize = W \times H$ $\sigma = 0.5$	82.71 %	86.10 %	**84.43 %**
$hsize = W \times H$ $\sigma = 0.25$	81.94 %	86.18 %	**84.08 %**
$hsize = W \times H$ $\sigma = 0.75$	82.33 %	86.34 %	**84.36 %**
$hsize = W \times H$ $\sigma = 1.0$	82.46 %	86.02 %	**84.25 %**

Table 4.4: Results For Image Level Gaussian Blurring With Various Parameters

4.1.3.1.1 Effect Of Image Level Blurring: Modifying the Filter Size (Averaging and Gaussian)

Using the averaging blurring filter, accuracy rates saw a definite decline as the filter size increased. This is reasonable behavior since filter size effectively controls the blur amount in this type of blurring as explained in Section 3.3.4.1 and seen in *Figure 3.12*. Moreover blurring with filter size $hsize = W \times H$ leads to a very visual effect when using the averaging filter as can be seen in *Figure 4.1*. As such, it makes sense that since the filter size is so strongly tied to the blur amount that accuracy decreases as filter size increases.

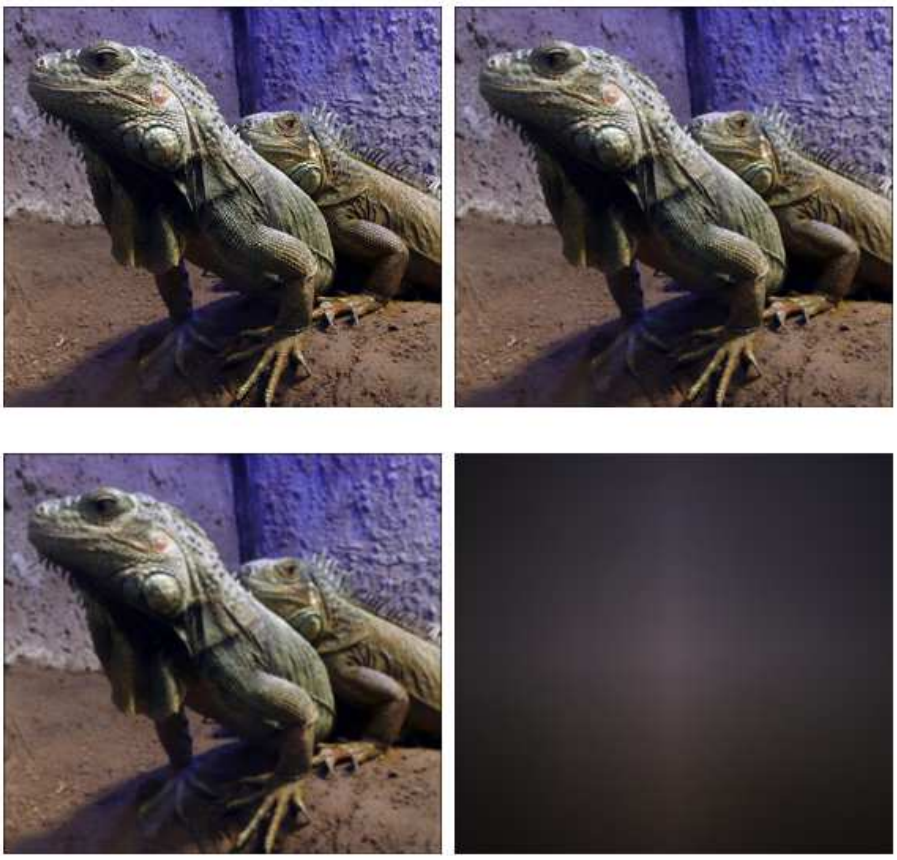

Figure 4.1: Averaging Filter On Au_Ani_10208.jpg (Top Left) With $hsize = 3 \times 3$ *(Top Right),* $hsize = 5 \times 5$ *(Lower Left), And* $hsize = W \times H$ *(Lower Right)*

The Gaussian blur approach on the other hand saw an increase in performance while increasing the filter size, achieving optimal performance with $hsize = W \times H$. Unlike when using the averaging filter, changing the filter size for the Gaussian blurring filter does not have as much of an impact on the perceived blur amount. Even with the largest filter size ($W \times H$) and the largest value for σ (1.0), the resulting image (*Figure 4.2*) is not affected nearly as much in comparison.

Figure 4.2: Gaussian Filter On Au_Ani_10208.jpg (Left) With $hsize = W \times H$, $\sigma = 1.0$ (Right)

4.1.3.1.2 Effect Of Image Level Blurring: Modifying σ (Gaussian)

The previous section led to the observation that the optimal filter size for Gaussian blurring in this application is equal to the dimensions of the image it is filtering. After experimenting with multiple values for σ, the optimal value was found to be the MATLAB-default of 0.5. Although the selection of $\sigma = 0.5$ is the best performer, its neighbors (0.25 and 0.75) perform similarly.

4.1.3.1.3 Effect of Image Level Blurring

Examining the results for the two types of blurring reveals that the Gaussian blurring filter outperforms the averaging blurring filter. The optimal configuration was achieved with a Gaussian blur filter with $hsize = W \times H$ and $\sigma = 0.5$. With the exception of the averaging filter with $hsize = W \times H$, all blurring was found to improve accuracy rates between 0.29% and 1.38%.

4.1.3.2 Experimental Results for Image Level Sharpening

Image level sharpening explores the possibility that edges should be accentuated in order to augment accuracy rates. This is achieved by sharpening, although with subtle amounts. The amount of sharpening is controlled with the parameter '`amount`'. Experimental results for various amounts are found in the table below:

Sharpening Amount	TPR	TNR	AR
amount = 0.25	82.68 %	83.97 %	**83.34 %**
amount = 0.5	82.29 %	84.33 %	**83.33 %**
amount = 0.75	81.77 %	84.41 %	**83.11 %**
amount = 1.0	81.08 %	84.94 %	**83.03 %**

Table 4.5: Results For Image Level Sharpening With Various Amounts

4.1.3.2.1 Effect of Image Level Sharpening

Image level sharpening proves here to be ineffective with accuracy rates decreasing as sharpening amount increases. Although the best performer of the bunch slightly outperforms the reference approach by 0.29%, it still does not outperform even the worst case of the image level Gaussian blurring configurations seen in *Table 4.4*.

4.1.4 Experimental Results for Content Adaptive Techniques

As was proven in Section 4.1.3, image level Gaussian blurring is an effective preprocessing method for improving the accuracy rate; much more so than image level sharpening. Although a constant set of blurring parameters still outperforms the reference approach by a significant margin, blurring based on the amount of higher frequency content is the focus of this section. Section 3.3.5 details a method for using image level DCT to approximate the amount of higher frequency content present in a given image which can be represented by a single representative value (μ). Section 4.1.4.1 and 4.1.4.2 both use this value μ as an input to a function that determines how much to blur a given image. While the former establishes two bins for processing based on a threshold μ value, the latter uses a linear function.

4.1.4.1 Experimental Results for the Binning Approach

Section 4.1.3.1 empirically determines that Gaussian lowpass filtering contributes the most to increased accuracy when $hsize = W \times H$ and $\sigma = 0.5$. The second highest performer has the same filter size but with $\sigma = 0.75$. Since these two are the top performers, they were selected as the two blur amounts for either side of the threshold.

The optimal choice for a threshold is not immediately evident and so some experimentation tried different values. Section 3.3.5.1 proposes a binning approach based off of percentile values for mu. 65^{th}, 70^{th}, 75^{th}, 80^{th}, and 90^{th} percentile values were chosen as possible thresholds. These percentiles map to thresholds as is seen in the table below:

Percentile	Authentic	Tampered	Combined
65th Percentile	$\mu = 9.52$	$\mu = 8.94$	$\mu = 9.25$
70th Percentile	$\mu = 10.04$	$\mu = 9.36$	$\mu = 9.75$
75th Percentile	$\mu = 10.65$	$\mu = 9.84$	$\mu = 10.25$
80th Percentile	$\mu = 11.30$	$\mu = 10.60$	$\mu = 11$
90th Percentile	$\mu = 13$	$\mu = 12.5$	$\mu = 12.75$

Table 4.6: Percentile Mappings To Threshold μ Values

Of course, there are two options once these blur amounts are chosen. Do smoother images (low-μ images) get blurred more or less? The table below examines which configuration performs better. Because a Gaussian blur filter with $hsize = W \times H$ was found to be the best performer in Section 4.1.3.1.1, this filter size is held constant in this adaptive testing. A threshold $\mu_{thresh} = 10.25$ is used for initial investigations.

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.75 - 0.5$ $\mu_{thresh} = 10.25$	82.37 %	85.50 %	**83.95 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ $\mu_{thresh} = 10.75$	82.82 %	85.82 %	**84.35 %**

Table 4.7: Results For Blurring Smoother Images More Versus Less

Table 4.7 shows that blurring images with greater amounts of higher frequency content with a larger σ to perform better. For this reason, the remainder of adaptive blurring configurations both in this section and Section 4.1.4.2 will also select larger σ values for images with greater amounts of higher frequency content.

With the determination that a larger σ should be used to blur images with greater amounts of higher frequency content, the next exploration seeks the optimal parameters for the binning function detailed in *Equation 4*. To choose the blur amount (σ) for the two bins, the best and second best performers were chosen from *Table 4.4*. This means that most of the images were blurred with $\sigma = 0.5$ while those with large amounts of high frequency content were blurred with $\sigma = 0.75$. The table below shows the impact of changing the threshold value of μ_{thresh} on accuracy rates:

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ $\mu_{thresh} = 9.25$	82.33 %	85.77 %	**84.07 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ $\mu_{thresh} = 9.75$	82.64 %	85.82 %	**84.26 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ $\mu_{thresh} = 10.25$	82.82 %	85.82 %	**84.35 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ $\mu_{thresh} = 11$	82.57 %	85.63 %	**84.12 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ $\mu_{thresh} = 12.75$	82.17 %	86.03 %	**84.12 %**

Table 4.8: Results For Choosing Different μ Thresholds

In *Table 4.8* it can be seen that selecting $\mu_{thresh} = 10.25$ yielded the best performance. However, it can also be said that the difference in σ values is sufficiently large such that a smaller span could be more effective. *Table 4.9* examines this possibility by using a smaller span:

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.5 - 0.625$ $\mu_{thresh} = 10.25$	82.43 %	85.71 %	**84.09 %**
$hsize = W \times H$ $\sigma = 0.625 - 0.75$ $\mu_{thresh} = 10.25$	82.12 %	86.35 %	**84.25 %**

Table 4.9: Results For Smaller Spans Of σ

4.1.4.1.1 Effect of Binning Approach

Performance using this configuration is somewhat strong with the optimal accuracy rate outperforming the reference approach by 1.3%. It was found that in this content adaptive context, images with greater degrees of higher frequency content should be blurred more (with larger values for σ). The selection of the two σs ($\sigma = 0.5$ and $\sigma = 0.75$) for the two bins was informed by the best and runner-up performer in *Table 4.4*. This initial selection was proved to be the best even when testing a reduced span of σ. Despite strong performance from this adaptive configuration, even the top performer failed to outperform blurring with a constant σ value across the authentic and tampered sets.

4.1.4.2 Experimental Results for the Linear Approach

While the binning approach detailed previously blurs images with only two different amounts based on amount of higher frequency content, the linear approach establishes a base blur amount and a modifier that assigns a different blur amount to each separate image based on the value for mu corresponding to the image. The calculation is described in *Equation 5*. Like in the binning approach, a range of blur amounts must be specified. Because the best and second best constant-σ performance was achieved with $\sigma = 0.5$ and $\sigma = 0.75$ respectively they are used as a starting point for the blurring span. *Table 4.7* indicates that images with greater degrees of higher frequency content should be blurred more and this observation is acknowledged in this approach as well. At first the entire span between 0.5 and 0.75 is used and then this is refined to two smaller regions. A Gaussian filter with $hsize = W \times H$ is maintained across all testing. Results are listed below:

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.5 - 0.75$	82.51 %	85.95 %	**84.25 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.625$	82.73 %	87.11 %	**84.93 %**
$hsize = W \times H$ $\sigma = 0.625 - 0.75$	81.83 %	86.03 %	**83.95 %**

Table 4.10: Results For Linearly Determined σ With Varying σ Spans

4.1.4.2.1 Effect of Linearly Determined σ

As can be seen in the table above, a linearly determined σ is indeed effective. At its best, accuracy rates are 1.88% better than the reference approach. Interestingly, a smaller span is found to be more effective for accuracy rates whereas this was not the case in the binning approach to content adaptive blurring. Another interesting point is that this content adaptive blurring configuration in fact outperforms a constant σ approach by 0.5%.

4.1.5 Experimental Results for Combining Preprocessing Techniques

In the sections previous, there is clear promise in using JPEG block line blurring and image level blurring to improve accuracy rates. Section 4.1.2 indicates the optimal number of neighbors and weights for JPEG block line blurring. Section 4.1.4.2 indicates the number optimal settings for image level blurring with content adaptive blur amounts determined by a linear function. The effect of combining these two enhancements will be shown in the subsequent tables. *Table 4.11* keeps the σ span constant at $\sigma = 0.5 - 0.625$. This was the optimal span determined in Section 4.1.4.2. *Table 4.12* and *Table 4.13* examine the other two spans also examined in Section 4.1.4.1 and Section 4.1.4.2 in the case that a combination of suboptimal parameters may exceed the optimal performers.

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.5 - 0.625$ 0.1, 0.2, 0.4, 0.2, 0.1	82.82 %	86.82 %	**84.84 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.625$ 0.25, 0.5, 0.25	82.88 %	86.65 %	**84.78 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.625$ 0.3, 0.4, 0.3	82.56 %	86.66 %	**84.61 %**

Table 4.11: Results For σ Span 0.5-0.625 With Varying JPEG Block Line Blurring Configurations

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.625 - 0.75$ 0.1, 0.2, 0.4, 0.2, 0.1	82.65 %	87.32 %	**85.01 %**
$hsize = W \times H$ $\sigma = 0.625 - 0.75$ 0.25, 0.5, 0.25	82.21 %	86.41 %	**84.33 %**
$hsize = W \times H$ $\sigma = 0.625 - 0.75$ 0.3, 0.4, 0.3	82.75 %	87.13 %	**84.95 %**

Table 4.12: Results For σ Span 0.625-0.75 With Varying JPEG Block Line Blurring Configurations

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ 0.1, 0.2, 0.4, 0.2, 0.1	82.72 %	86.67 %	**84.71 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ 0.25, 0.5, 0.25	82.93 %	87.01 %	**84.99 %**
$hsize = W \times H$ $\sigma = 0.5 - 0.75$ 0.3, 0.4, 0.3	82.71 %	86.61 %	**84.67 %**

Table 4.13: Results For σ Span 0.5-0.75 With Varying JPEG Block Line Blurring Configurations

Combining JPEG block line blurring with linear adaptive σ is the obvious first choice but it is also possible that a constant σ will perform better when combined with JPEG block line blurring. In *Table 4.14* the JPEG block line blurring configuration is kept constant but the top two performers from *Table 4.4* are selected for a constant blur amount across all of the images in the authentic and tampered sets.

Parameters	TPR	TNR	AR
$hsize = W \times H$ $\sigma = 0.5$ 0.1, 0.2, 0.4, 0.2, 0.1	82.55 %	86.70 %	**84.65 %**
$hsize = W \times H$ $\sigma = 0.75$ 0.1, 0.2, 0.4, 0.2, 0.1	82.78 %	86.96 %	**84.89 %**

Table 4.14: Results For Keeping JPEG Block Line Blurring Constant With Varying Constant σ Values

4.1.5.1 Effect of Combining JPEG Block Line Blurring and Image Level Blurring

Upon examining *Table 4.11*, results are initially discouraging with even its best performer failing to outperform linear content adaptive blurring by itself. Since *Table 4.11* represents the combination of optimal JPEG block line blurring configuration and the optimal linear adaptive blurring configuration, the viability of this approach is questioned. However, combining a suboptimal content adaptive σ span with the optimal JPEG block line (*Table 4.12*) blurring configuration does lead to another gain of 0.08% over linear content adaptive blurring on its own. This puts overall gain at 1.98% over the reference approach and the accuracy rate above 85%.

4.2 CLASSIFIER BIAS TOWARDS IDENTIFYING TAMPERED IMAGES

Examining the results in Section 4.1 as a whole reveals an interesting pattern in detection rates. Despite providing an equal number of tampered and authentic feature vectors to the machine learning classifier, the true negative rate (TNR) is consistently higher than the true positive rate (TPR) by about 3-5%. Put another way, the extracted features in this framework lead to more accurate classification rates for tampered images than for authentic images.

When examining reported results from some detection frameworks noted in Section 2.1.2, not all studies indicate TPR, TNR, and AR separately; however those that do also see this gap. [18], [22], [23], and [33] all report absolute differences between TPR and TNR between 2-28%. Interestingly, these studies are not in agreement about whether correct identification of tampered images outperforms correct identification of authentic

images or vice versa. Various image splicing datasets are utilized among these four publications, which suggests that the introduced gap cannot be attributed to this variable.

This gap has not been addressed in any of these publications which leaves the cause of the problem open to discussion. An intuitive guess may point to the likelihood that tampered edges are made up of higher frequency content and that the classifier consequently struggles to correctly identify authentic images with higher frequency content. To see if this is true, results for the reference, JPEG block line blurring combined with adaptive blurring, JPEG block line blurring combined with constant blurring, and adaptive blurring were all examined. Looking at the top Nth percentile (80, 90, 95, 99) of images with higher frequency content shows that as images become "busier" the classifier tends to improperly bin authentic images more frequently than tampered images. This can be seen in the figures below:

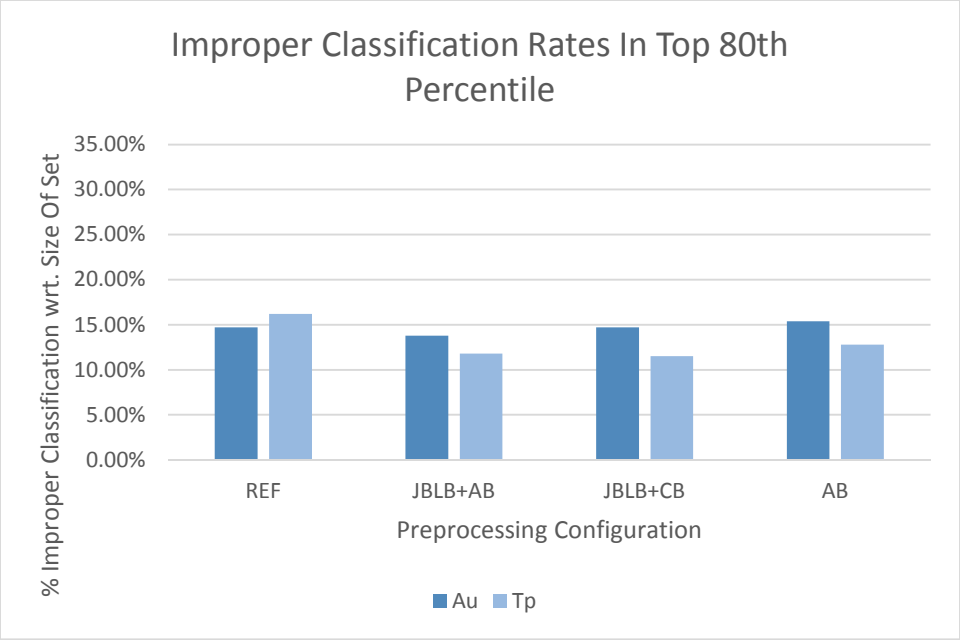

Figure 4.3: Improper Classification Rates In Top 80th Percentile

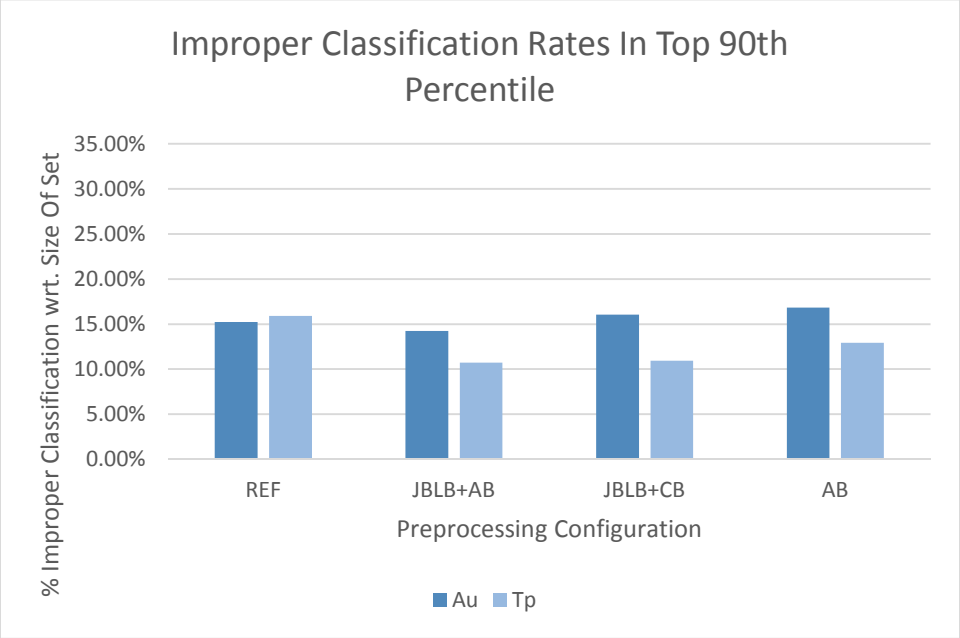

Figure 4.4: Improper Classification Rates In Top 90th Percentile

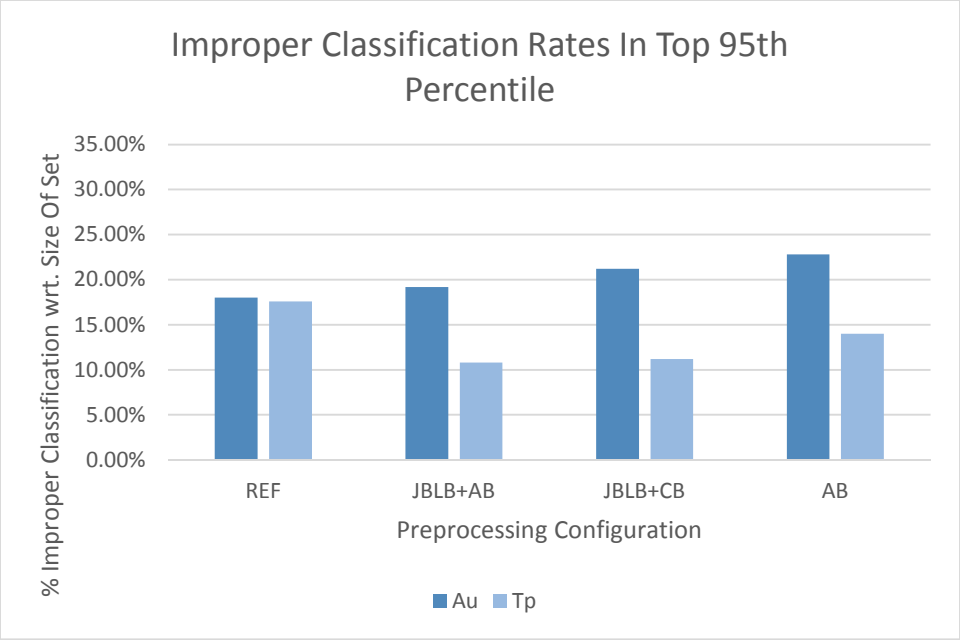

Figure 4.5: Improper Classification Rates In Top 90th Percentile

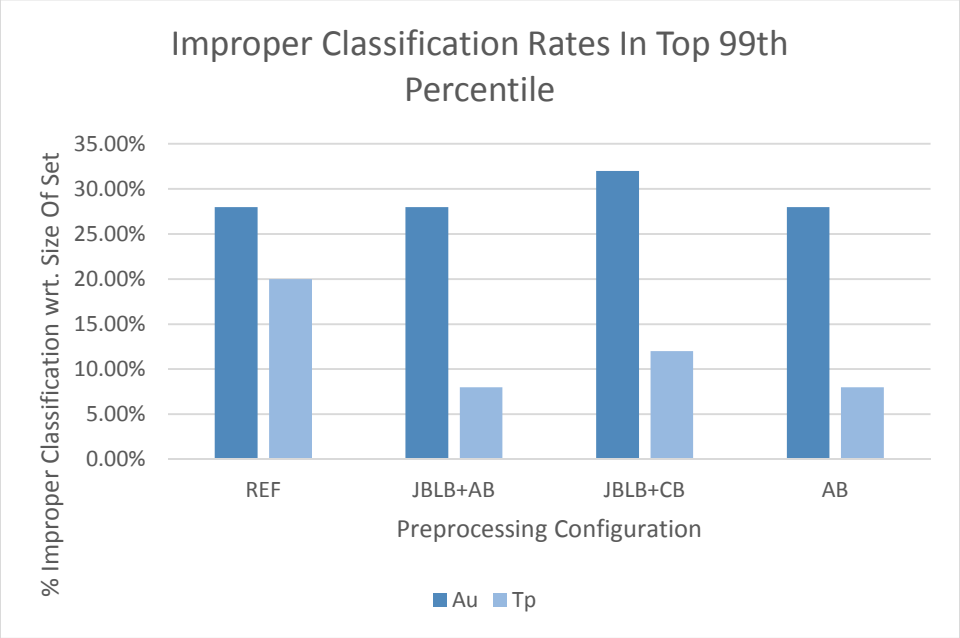

Figure 4.6: Improper Classification Rates In Top 99th Percentile

It is interesting to see that in every case but the reference, authentic images are improperly classified more often and by a significant margin as different percentile bins are examined. This pattern appears to support the hypothesis that the model created from authentic and tampered feature vectors causes the classifier to struggle to properly identify authentic images with higher frequency content.

4.3 Experimental Results for Divergent Quality Factor Detection

Subjecting these different datasets to different JPEG quality factors translates to a specific set of quantization tables. These quantization tables effectively bin values along the 2D DCT traversal pattern in *Figure 3.16* with increasing aggressiveness as the lower right hand corner is reached. Reduction in fidelity (introduction of lossiness) is tied to the quality factor, which chooses which quantization table will be used. Lower quality factors mean lossier quantizations. The luminance and chrominance information is subjected to different sets of quantization tables, leveraging the fact that chrominance information can be compressed more, due to human vision system intricacies (greater sensitivity to changes in light intensity than to changes in color).

In testing prior to this section, the authentic and tampered sets in the CASIA TIDE database v2 have already been subjected to JPEG quality factor equalization as described in Section 3.3.1.

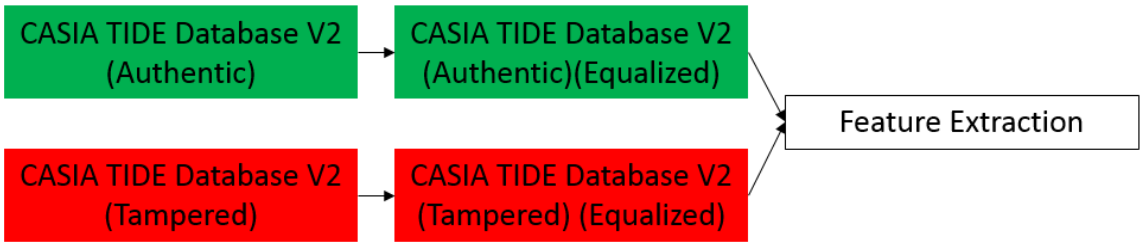

Figure 4.7: Project Structure Thus Far

As a continuation of that, this examination subjects the tampered set to one more pass of JPEG processing with a variety of quality factors. Then, features are extracted from these reprocessed version of the tampered set to examine the impact of the degree of quality factor difference between the authentic and tampered sets on detection rates. Quality factors of 70, 80, 84, 90, and 95 were applied to the tampered set to test this relationship. *Figure 4.8* shows one instance of the testing to be done in this section (Au 84, Tp 84->70) but there are 4 other instances – one for each reprocessed version of the tampered set.

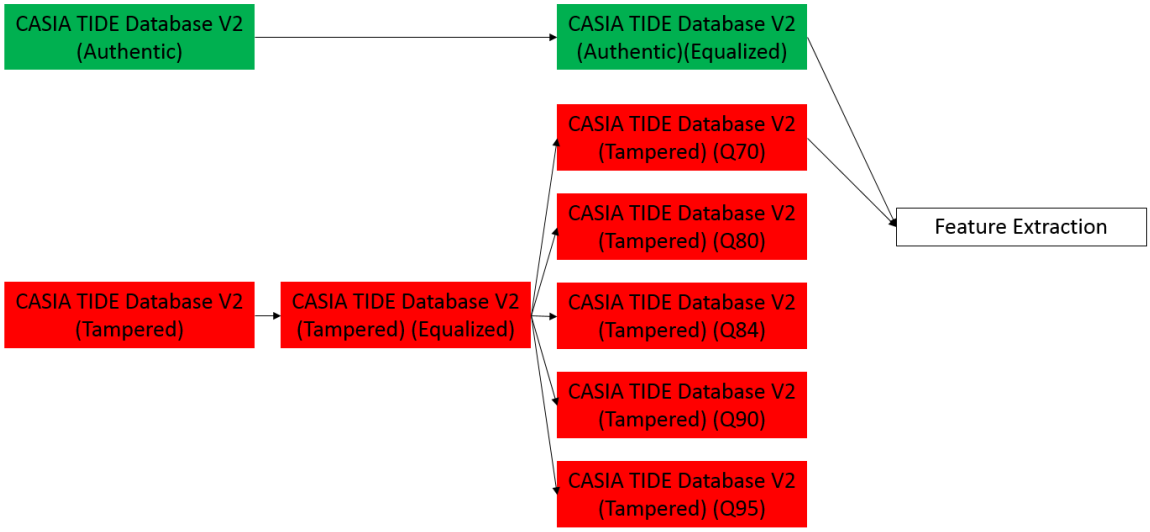

Figure 4.8: Testing Structure For This Section

The reference approach, the highest performer, and two of the second highest performers were chosen to examine this relationship to determine if one approach expressed a particular tolerance for a mismatch in quality factors. In the table below, "Au" indicates the authentic set while "Tp" indicates the tampered data set.

Au/Tp Pairing	Reference	0.1, 0.2, 0.4, 0.625-0.75	0.1, 0.2, 0.4 0.75	0.5-0.625
Au 84, Tp 84->70	98.45 %	98.46 %	98.5 %	98.5 %
Au 84, Tp 84->80	96.64 %	94.53 %	95.04 %	98.42 %
Au 84, Tp 84->84	79.46 %	80.91 %	80.68 %	80.53 %
Au 84, Tp 84->90	96.05 %	90.5 %	90.27 %	91.92 %
Au 84, Tp 84->95	95.59 %	89.4 %	89.68 %	91.3 %

Table 4.15: Results For Different Quality Factor Mismatches

One obvious trend across results for the different approaches is that subjecting the tampered set to another pass of Q=84 compression provides the worst results when comparing it with an authentic set also compressed with Q=84. This is likely because this quality factor is the same as the quality factor done in the authentic set compression and reuse of the same quality factor coefficients makes the features less distinguishable. As the quality factor moves away (in both the positive and negative directions) accuracy rate increases sharply as the result of different quantization occurring which impacts the extracted features.

Table 4.16 analyzes these results with respect to both average and variance:

Measure	Reference	0.1, 0.2, 0.4, $\sigma = 0.625 - 0.75$	0.1, 0.2, 0.4 0.75	0.5-0.625
Average	93.24 %	91.10 %	90.83 %	91.41 %
Variance	48.43	42.53	36.24	36.11

Table 4.16: Statistics For Varying Quality Factors

This thesis works off of the claim that using the CASIA TIDE database V2 is not the most representative of a real world situation. Nonetheless, there is also value in understanding the performance of these preprocessing techniques on the authentic and tampered datasets as-is. Again, the top performing approaches from *Table 4.15* are compared in *Table 4.17* to examine the impact of these preprocessing techniques on un-equalized datasets.

Au/Tp Pairing	Reference	0.1, 0.2, 0.4, 0.625-0.75	0.1, 0.2, 0.4 0.75	0.5-0.625
Au Orig/Tp Orig	96.54 %	95.28 %	95.18 %	95.42 %

Table 4.17: Results For Top Performing Approaches Using Un-Equalized CASIA TIDE Database V2

4.3.1 Effect of Divergent Quality Factor Detection

Table 4.16 compares average and variance statistics for each of the approaches listed in *Table 4.15.* A comparison of these statistics reveals that the reference approach has the best average performance across the various datasets. However, in the worst case

scenario (Au 84, Tp 84->84) the combination of JPEG block line blurring and linear content adaptive image level blurring is the best performer, as was seen in Section 4.1. If variance can be used as a measure of consistency, the strictly content adaptive approach and the JPEG block line blurring paired with a constant σ both performed similarly. Another interesting perspective on the data in *Table 4.15* is average and variance statistics when the worst case is excluded:

Measure	Reference	0.1, 0.2, 0.4, $\sigma = 0.625 - 0.75$	0.1, 0.2, 0.4 0.75	0.5-0.625
Average	96.68 %	93.22 %	93.37 %	94.14 %
Variance	1.18	12.79	13.08	8.12

Table 4.18: Statistics For Varying Quality Factors (Worst Case Omitted)

Table 4.18 examines the effectiveness of the various approaches in situations where the authentic and tampered datasets are more separated in terms of processing associated with different quality factors. Under these circumstances, not only is the reference approach the top performer with respect to the average but also with respect to variance. This indicates that the reference method is actually more tolerant in situations where there is separation in quality factors between the authentic and tampered sets.

This point is enforced by the findings in *Table 4.17* which shows the average accuracy rate of the top performers on the authentic and tampered sets which have not been subjected to quality factor equalization.

5 CONCLUSION

This thesis proposes the inclusion of preprocessing techniques into future image splicing detection frameworks. Blurring along JPEG block lines (+1.28%) and image level blurring (+1.3%) were both found to increase accuracy rates themselves but combining the two boosted accuracy even further (+1.84%). Choosing blur amounts based on amounts of higher frequency content proved to be effective as well (+1.88%). The optimal configuration of preprocessing techniques and its parameters led to a 1.98% gain in accuracy over the reference framework when using the authentic and tampered datasets from the CASIA TIDE database v2 with quality factor equalization applied to them.

This thesis also addresses a bias inherent to detection between authentic and tampered JPEG content that does not see acknowledgement in modern publications. By accounting for this bias and equalizing JPEG quality factor gaps, this thesis ensures that it is extracting and classifying features in the toughest use case. By exploring how the accuracy rate responds to a varying tampered set quality factor it was shown that the reference approach outperforms any tested combination of preprocessing approaches when excluding the worst case from consideration. However, in the worst case, the same combination of JPEG block line blurring and linear content adaptive blurring proved to be the top performer.

Of course in the real world it would be naïve to expect authentic and test sets to be of uniform quality factor. An area for future work could be to devise a heterogeneous quality factor assignment across the authentic and tampered sets for a more realistic analysis. An even more ambitious solution would be to devise a sufficiently diverse and

challenging dataset comprised only of images that have never been compressed. This is challenging from many perspectives however (e.g. content acquisition, skilled tampering efforts, file size of uncompressed formats impacting distribution)

Because this thesis proposes a number of preprocessing techniques shown to positively impact the reference framework, it is believed that these techniques will positively affect other existing splicing detection frameworks. This could be another area of future work.

Finally, although an initial hypothesis has been proposed for the gap in true positive rates and true negative rates for this particular framework, it is another subject that can be expanded in further research. While for this project it is suggested that the classifier struggles to properly label authentic images with greater degrees of higher-frequency content due to patterns in higher frequencies of tampered images, a more generalized or accurate answer may exist.

6 REFERENCES

1 J. Dong and W. Wang, "CASIA Tampered Image Detection Evaluation Database (CASIA TIDE v2.0)," Chinese Academy of Sciences, 2010. [Online]. Available: http://forensics.idealtest.org:8080/index_v2.html.

2 S. Goldenberg, "Take one part Kerry, one part Fonda ...," 18 February 2004. [Online]. Available: http://www.theguardian.com/media/2004/feb/18/newmedia.uselections2004.

3 M. Mishra and F. L. D. M. C. Adhikary, "Digital Image Tamper Detection Techniques - A Comprehensive Study," *International Journal of Computer Science and Business Informatics,* vol. 2, no. 1, p. 12, 2013.

4 The Washington Times, "Photo of Kerry with Fonda enrages Vietnam veterans," 11 February 2004. [Online]. Available: http://www.washingtontimes.com/news/2004/feb/11/20040211-123002-8027r/?page=all.

5 IEEE, "IEEE Xplore Digital Library," [Online]. Available: http://ieeexplore.ieee.org/. [Accessed 18 January 2015].

6 M. Wu and B. Liu, "Watermarking for image authentication," *Image Processing, 1998. ICIP 98. Proceedings. 1998 International Conference on ,* vol. 2, pp. 437-441, 1998.

7 D. Artz, "Digital steganography: hiding data within data," *Internet Computing, IEEE,* vol. 5, no. 3, pp. 75-80, 2001.

8 J. A. Redi, W. Taktak and J.-L. Dugelay, "Digital Image Forensics: a booklet for beginners," *Multimedia Tools and Applications,* 2010.

9 P. Blythe and J. Fridrich, "Secure Digital Camera," *Proceedings of Digital Forensic Research Workshop (DFRWS),* 2004.

10 A. Joshi, A. Darji and V. Mishra, "Design and implementation of real-time image watermarking," *Signal Processing, Communications and Computing (ICSPCC), 2011 IEEE International Conference on,* pp. 1-5, 2011.

11 C. Rey and J.-L. Dugelay, "A survey of watermarking algorithms for image authentication," *EURASIP Journal on Applied Signal Processing,* no. 6, pp. 613-621, 2002.

12 C.-C. Chang and C.-J. Lin, "LIBSVM -- A Library for Support Vector Machines," *ACM Transactions on Intelligent Systems and Technology,* vol. 2, no. 3, pp. 27:1--27:27, 2011.

13 T. K. Huynh, K. V. Huynh, T. Le-Tien and S. C. Nguyen, "A Survey on Image Forgery Detection Techniques," *Computing & Communication Technologies - Research, Innovation, and Vision for the Future (RIVF), 2015 IEEE RIVF International Conference on,* pp. 71-76, 2015.

14 A. C. Popescu and H. Farid, "Exposing Digital Forgeries by Detecting Traces of Resampling," *IEEE TRANSACTIONS ON SIGNAL PROCESSING,* vol. 53, no. 2, pp. 758-767, 2005.

15 B. Mahdian and S. Saic, "Blind Authentication Using Periodic Properties of Interpolation," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 3, no. 3, pp. 529-538, 2008.

16 B. Mahdian and S. Saic, "Detection of Resampling Supplemented with Noise Inconsistencies Analysis for Image Forensics," *International Conference on Computational Sciences and Its Applications,* pp. 546-556, 2008.

17 T.-T. Ng, S.-F. Chang and Q. Sun, "Blind Detection of Photomontage Using Higher Order Statistics," *Circuits and Systems, 2004. ISCAS '04. Proceedings of the 2004 International Symposium on,* vol. 5, pp. 688-691, 2004.

18 P. Sutthiwan, Y. Q. Shi, H. Zhao, T.-T. Ng and W. Su, "Markovian Rake Transform for Digital Image Tampering Detection," *Transactions on DHMS VI,* pp. 1-17, 2011.

19 Z. Kaizhen and Z. Zhang, "A Novel Algorithm Of Image Splicing Detection," *2012 International Conference on Industrial Control and Electronics Engineering,* pp. 1927-1930, 2012.

20 A. A. Alahmadi, M. Hussain, H. Aboalsamh, G. Muhammad and G. Bebis, "Splicing Image Forgery Detection Based on DCT and Local Binary Pattern," *GlobalSIP 2013,* pp. 253-256, 2013.

21 M. Hussain, G. Muhammad, S. Q. Saleh, A. M. Mirza and G. Bebis, "Image Forgery Detection Using Multi-Resolution Weber Local Descriptor," *EuroCon 2013,* pp. 1570-1577, 2013.

22 W. Wang, J. Dong and T. Tan, "Image Tampering Detection Based on Stationary Distribution of Markov Chain," *IEEE ICIP 2010,* pp. 2101-2104, 2010.

23 X. Zhao, S. Wang, S. Li, J. Li and Q. Yuan, "Image Splicing Detection Based on Noncausal Markov Model," *Image Processing (ICIP), 2013 20th IEEE International Conference on,* pp. 4462-4466, 2013.

24 J. Hou, H. Shi, Y. Cheng and R. Li, "Fogery Image Splicing Detection by Abnormal Prediction Features," *Mechatronics and Automation (ICMA), 2013 IEEE International Conference on,* pp. 1394-1398, 2013.

25 O. Muratov, D.-T. Dang-Nguyen, G. Boato and F. G. D. Natale, "Saliency Detection As A Support For Image Forensics," *Communications Control and Signal Processing (ISCCSP), 2012 5th International Symposium on,* pp. 1-5, 2012.

26 Q. Zheng, W. Sun and W. Lu, "Digital Spliced Image Forensics Based on Edge Blur Measurement," *Information Theory and Information Security (ICITIS), 2010 IEEE International Conference on,* pp. 399-402, 2010.

27 M. Doyoddorj and K.-H. Rhee, "A Blind Forgery Detection Scheme Using Image Compatability Metrics," *Industrial Electronics (ISIE), 2013 IEEE International Symposium on,* pp. 1-6, 2013.

28 H. P, L. S. Nair, A. S.M, R. Unni, V. P. H and P. Poornachandran, "Digital Image Forgery Detection on Artificially Blurred Images," *Emerging Trends in Communication, Control, Signal Processing & Computing Applications (C2SPCA), 2013 International Conference on,* pp. 1-5, 2013.

29 K. Bahrami and A. C. Kot, "Image Tampering Detection By Exposing Blur Type Inconsistency," *Acoustics, Speech and Signal Processing (ICASSP), 2014 IEEE International Conference on,* pp. 2654-2658, 2014.

30 K. Bahrami, A. C. Kot and J. Fan, "Splicing Detection in Out-of-Focus Blurred Images," *Information Forensics and Security (WIFS), 2013 IEEE International Workshop on,* pp. 144-149, 2013.

31 P. Kakar, N. Sudha and W. Ser, "Exposing Digital Image Forgeries by Detecting Discrepancies in Motion Blur," *IEEE TRANSACTIONS ON MULTIMEDIA,* vol. 13, no. 3, pp. 443-452, 2011.

32 "Image Tamper Detection Baed on Demosaicing Artifacts," *Image Processing (ICIP), 2009 16th IEEE International Conference on,* pp. 1497-1500, 2009.

33 Z. Fang, S. Wang and X. Zhang, "Image Splicing Detection Using Camera Characteristic Inconsistency," *International Conference on Multimedia Information Networking and Security,* pp. 20-24, 2009.

34 Y.-F. Hsu and S.-F. Chang, "Camera Response Functions for Image Forensics: An Automatic Algorithm for Splicing Detection," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 5, no. 4, pp. 816-825, 2010.

35 L. Yang, X. Zhang and J. Ren, "Precision Detection of CCD Splicing Based on Template matching Algorithm," *Computing, Control and Industrial Engineering (CCIE), 2011 IEEE 2nd International Conference on,* vol. 2, pp. 224-227, 2011.

36 Y. Dai, H. Xiang, P. Lu and W. Feng, "Image Splicing Detection Based on Estimation of Camera Intrinsic Parameters," *Audio Language and Image Processing (ICALIP), 2010 International Conference on,* pp. 214-218, 2010.

37 S. Tiago José de Carvalho, C. Riess, E. Angelopoulou, H. Pedrini and M. I. Anderson de Rezende Rocha, "Exposing Digital Image Forgeries by Illumination Color Classification," *IEEE TRANSACTIONS ON INFORMATION FORENSICS AND SECURITY,* vol. 8, no. 7, pp. 1182-1194, 2013.

38 X. Wu and Z. Fang, "Image Splicing Detection Using Illuminant Color Inconsistency," *Multimedia Information Networking and Security (MINES), 2011 Third International Conference on,* pp. 600-603, 2011.

39 J. Hsu and S.-F. Chang, "Columbia Image Splicing Detection Evaluation Dataset," Columbia DVMM Research Lab, 2004. [Online]. Available: http://www.ee.columbia.edu/ln/dvmm/downloads/AuthSplicedDataSet/AuthSplicedDataSet.htm.

40 Biodiversity Sciences Technology, "CalPhotos," University of California, Berkeley, 2000. [Online]. Available: http://calphotos.berkeley.edu//.

41 J. Hsu and S.-F. Chang, "Columbia Uncompressed Image Splicing Detection Evaluation Dataset," Columbia DVMM Research Lab, 2006. [Online]. Available: http://www.ee.columbia.edu/ln/dvmm/downloads/authsplcuncmp/.

42 J. Dong and W. Wang, "CASIA Tampered Image Detection Evaluation Database (CASIA TIDE v1.0)," Chinese Academy of Sciences, 2010. [Online]. Available: http://forensics.idealtest.org:8080/index_v1.html.

43 J. Dong, W. Wang and T. Tan, "CASIA IMAGE TAMPERING DETECTION EVALUATION DATABASE," *Signal and Information Processing (ChinaSIP),* pp. 422-426, 2013.

44 G. Schaefer and M. Stich, "UCID - An Uncompressed Colour Image Database," *Society of,* vol. 5307, pp. 472-480, 2004.

45 MathWorks, "randperm: random permultation," 2015. [Online]. Available: http://www.mathworks.com/help/matlab/ref/randperm.html?refresh=true. [Accessed 14 February 2015].

46 MathWorks, "imwrite: Write image to graphics file," 2015. [Online]. Available: http://www.mathworks.com/help/matlab/ref/imwrite.html?refresh=true. [Accessed 15 February 2015].

47 X. Li, B. Gunturk and L. Zhang, "Image Demosaicing: a Systematic Survey," *SPIE Visual Communications and Image Processing 2008,* vol. 6822, 2008.

48 MathWorks, "rgb2ycbcr: Convert RGB color values to YCbCr color space," [Online]. Available: http://www.mathworks.com/help/images/ref/rgb2ycbcr.html?refresh=true. [Accessed 29 January 2015].

49 International Telecommunications Union, "STUDIO ENCODING PARAMETERS OF DIGITAL TELEVISION FOR STANDARD 4:3 AND WIDE-SCREEN 16:9 ASPECT RATIOS," 1995. [Online]. Available: https://www.itu.int/dms_pubrec/itu-r/rec/bt/R-REC-BT.601-5-199510-S!!PDF-E.pdf.

50 J. D. a. T. T. Wei Wang, "Effective Image Splicing Detection Based on Image Chroma," *IEEE ICIP 2009,* pp. 1257-1260, 2009.

51 B. Oztana, A. Malik, Z. Fan and R. Eschbach, "Removal of Artifacts from JPEG Compressed Document Images," *Color Imaging XII: Processing, Hardcopy, and Applications,* vol. 6493, 2007.

52 Z.-N. Li, "4.2. Image Compression -- JPEG," Simon Fraser University, [Online]. Available: http://www-i6.informatik.rwth-aachen.de/web/Misc/Coding/365/li/material/notes/Chap4/Chap4.2/Chap4.2.html. [Accessed 27 March 2015].

53 MathWorks, "fspecial: Create predefined 2D Filter," 2015. [Online]. Available: http://www.mathworks.com/help/images/ref/fspecial.html?refresh=true. [Accessed 16 February 2015].

54 MathWorks, "imfilter: N-D Filtering of Multidimensional Images," 2015. [Online]. Available: http://www.mathworks.com/help/images/ref/imfilter.html. [Accessed 16 February 2015].

55 MathWorks, "imsharpen: Sharpen an Image Using Unshap Masking," [Online]. Available: http://www.mathworks.com/help/images/ref/imsharpen.html. [Accessed 31 January 2015].

56 Y. Wang, "DCT and Transform Coding," Polytechnic University, 2006. [Online]. Available: http://eeweb.poly.edu/~yao/EE3414/ImageCoding_DCT.pdf.

57 MathWorks, "prctile: Percentiles of a data set," 2015. [Online]. Available: http://www.mathworks.com/help/stats/prctile.html?refresh=true. [Accessed 19 Feubrary 2015].

58 S. R. Gunn, "Support Vector Machines for Classification and Regression," *University of Southampton,* pp. 1-66, 1998.

59 T. Hastie, R. Tibshirani and J. Friedman, "7.9 Vapnik-Chervonenkis Dimension," in *The Elements of Statistical Learning: Data Mining Interference and Prediction*, New York, Springer, 2009, p. 238.

60 MathWorks, "confusionmat: Confusion matrix," 2015. [Online]. Available: http://www.mathworks.com/help/stats/confusionmat.html?refresh=true. [Accessed 18 February 2015].

61 MathWorks, "crossvalind: Generate cross-validation indices," 2015. [Online]. Available: http://www.mathworks.com/help/bioinfo/ref/crossvalind.html?searchHighlight=crossvalind. [Accessed 18 February 2015].