UNIVERSITY OF CALIFORNIA SAN DIEGO

Real-Time Contrast Enhancement for 3D Medical Image Stacks

A thesis submitted in partial satisfaction of the requirements
for the degree Master of Science

in

Computer Science

by

Karen Lucknavalai

Committee in charge:

Professor Jürgen Schulze, Chair
Professor Ravi Ramamoorthi
Professor Larry Smarr

2020

The Thesis of Karen Lucknavalai is approved, and it is acceptable in quality and form for publication on microfilm and electronically:

_____

_____

_____

Chair

University of California San Diego

2020

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

ACKNOWLEDGEMENTS

I would like to thank my friends and family for their support and continual words of encouragement through this process, especially to my main support system, my husband, without whom I would not be where I am today.

A big thank you to my advisor, Jürgen Schulze whose patience and guidance throughout the years have been invaluable to me. Thanks as well to the entire Immersive Visualization Lab who were also incredibly helpful throughout this work.

VITA

2010	B. S. in Mathematics *cum laude*, University of California Irvine
2010	B. F. A. in Dance Performance *cum laude*, University of California Irvine
2018-2020	Graduate Teaching Assistant, University of California San Diego
2020	M. S. in Computer Science, University of California San Diego

PUBLICATIONS

Menghe Zhang, Karen Lucknavalai, Weichen Liu, Kamran Alipour, and Jurgen P. Schulze, "ARCalVR: Augmented Reality Playground on Mobile Devices", *ACM SIGGRAPH AppyHour*, 2019.

Zhang, M., Lucknavalai, K., Liu, W., Schulze, J.P., "CalAR: A C++ Engine for Augmented Reality Applications on Android Mobile Devices", *In Proceedings of IS&T The Engineering Reality of Virtual Reality*, San Francisco, CA, January 30, 2020

ABSTRACT OF THE THESIS

Real-Time Contrast Enhancement for 3D Medical Image Stacks

by

Karen Lucknavalai

Master of Science in Computer Science

University of California San Diego, 2020

Professor Jürgen Schulze, Chair

Medical professionals rely on Medical Imaging to help diagnose and treat patients. It is therefore important for them to be able to see all the details captured in the images. Often the use of contrast enhancement or noise reduction techniques are used to help improve the image quality. This thesis introduces a real-time implementation of 3D Contrast Limited Adaptive Histogram Equalization to enhance 3D medical image stacks, or volumes. This algorithm can be used interactively by medical doctors to help visualize the 3D medical volumes and prepare for surgery. It also introduces two novel extensions to the algorithm to allow a user to interactively decide what region to focus the enhancement, Focused CLAHE and Masked CLAHE. Focused CLAHE applies the 3D CLAHE algorithm to a specified block of the entire medical volume and

Masked CLAHE applies the algorithm to a selected organ or organs. These three contributions can be used, to not only help improve the visualization of 3D medical image stacks, but also to provide that contrast enhancement in real-time.

Chapter 1

Introduction

1.1 Motivation

Medical imaging is critical when it comes to helping doctors diagnose patients, and prepare for surgeries. Currently, medical professionals are viewing and studying these scans one slice at a time on 2D monitors. Given that this is a considerable difference from the reality they experience in an actual surgery, it seems advantageous to be able to view and interact with this 3D data in a 3D environment.

The goal of this thesis is to develop and implement an algorithm that would help improve the contrast of MRI data on an 8 bit display within a Virtual Reality application. In order to accomplish this the algorithm needs to improve the contrast of the medical volumes, and efficiently map the 10-12 bits of data present in the medical volumes to 8 bits for the display. In addition, the algorithm must be implemented efficiently in order to make sure that the interaction of the application within a Virtual Reality Environment can be done in real-time.

1.2 HDR Imaging

Dynamic Range is the range of numbers that are available to represent colors and gray values in an image. Standard Dynamic Range or Low Dynamic Range uses 8 bits per pixel per channel, also referred to as 8 bits per sample. This means that the red, green and blue values for the image can vary between $[0, 2^8 - 1]$ or equivalently $[0, 255]$. Most images are captured with and viewed on screens that use Standard Dynamic Range (SDR). In most cases SDR is sufficient to capture the details of a scene. However, if the scene contains a wider range of brightness values than this $[0, 255]$ range, there will be parts of the image without the original detail that appear washed out or too dark. The washed out areas are over exposed, while the areas that are too dark are under exposed. Photographers can capture the detail in these different regions of the scene by increasing or decreasing the exposure. While changing the exposure will change the regions of the captured image that are properly exposed, it will not solve the underlying problem that the scene contains a wider range of brightness than the Standard Dynamic Range can capture.

In order to properly capture the detail in both the over and under exposed areas, the dynamic range needs to be increased. High Dynamic Range images do exactly that. High Dynamic Range (HDR) is simply a dynamic range that is greater than the Standard Dynamic Range. HDR images usually store values that range between 10 and 16 bits per sample. This increase in dynamic range makes it possible to capture and display details in the pixels that were either over or under exposed in a Standard Dynamic Range image. The difference between a SDR and HDR images can be seen in Figure 1.1. The top row contains SDR images, and the bottom row are their HDR counterparts. Notice that areas that appear shadowed or dark in the SDR images become more detailed in their corresponding HDR images. This is the result of that extended dynamic range.

Creating HDR images is generally done by combining multiple images of the same scene taken at different exposures. This idea was first introduced to take images of seascapes in the

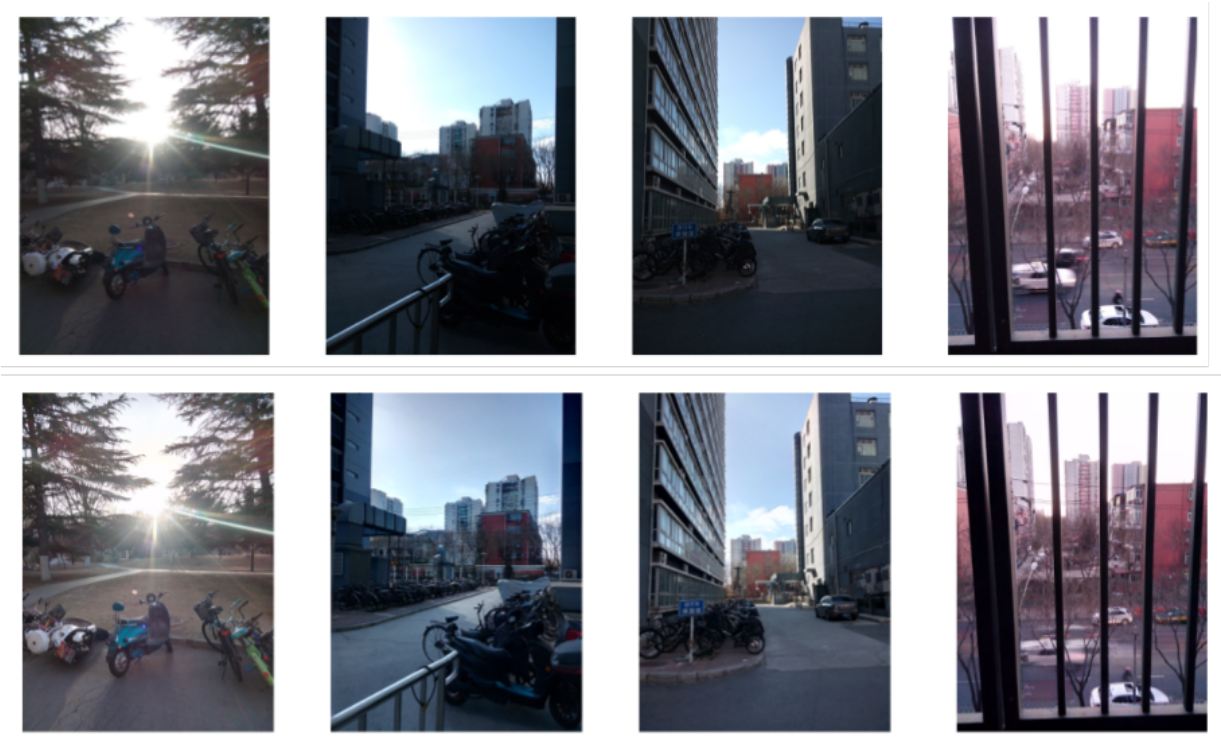


Figure 1.1: Comparison of SDR and HDR images. The SDR images are in the top row, and their corresponding HDR images are in the bottom row. These images are from [1]

1850s. One image was taken to capture the sky and another was taken at a different exposure for the sea. The two negatives were then combined together at the horizon to create the final image. In the 1990's many more methods were developed to capture HDR images, as well as the development of the mathematical theory of differently exposed pictures. As digital cameras became more available these HDR algorithms similarly increased in demand. Today most smart phones come equipped with the ability to capture HDR images, and it is done using that same original idea of combining multiple pictures taken at different exposures.

Since increasing the dynamic range allows for more detail to be captured in the final image, why not always use HDR imaging? Capturing the detail in an image is just half of it. In order to view these images, they either need to be printed, or displayed on a screen. Both the colors that can be printed and the bits of data that can be displayed on screens are limited. While HDR screens do exist, they are uncommon. Most screens can only display 8 bits of data. So even

if the HDR image captured 16 bits of detail, it is not possible to fully display that data without a special HDR monitor.

1.3 Medical Imaging

Medical Imaging began back in 1895 with the invention of the x-ray. X-rays use radiation to produce images, usually of bones or teeth. These images are generated by shooting X-rays at the desired body part. Different tissues absorb these rays at different amounts, and the rays that pass through the body create the x-ray image. In the 1960's sonar technology led to the development of ultrasound machines which send high frequency sound waves through a probe. When the sound waves bounce back, the probe can detect the different structures inside the body which are then converted into images. Computed Tomography (CT) scans were developed in the 1970's and are similar to taking an x-ray at a cross-section of the body. These cross-section images are generally referred to as slices and are able to pick up more subtle detail that an x-ray would be able to. Magnetic Resonance Imaging (MRI) was developed around the same time as CT scans, and also provide cross-sectional images of the body. However MRI's use magnets and radio waves to help create more detailed images than CT's are able to produce. MRI's work by using those radio waves to excite the protons in the body, and pull them out of equilibrium with the magnetic field. When the radio waves are turned off, MRI sensors detect the amount of energy released as the protons realign with the magnetic field. The amount of energy released, as well as the time it takes for the protons to realign, depends on the type of tissue. Examples of the difference between CT and MRI scans can be seen in Figure 1.2 [11]. While different types of imaging are better suited to different circumstances, they all have the goal of helping medical professionals diagnose patients, monitor conditions, or help plan for treatment and surgeries. It is the hope that with the proper imaging, doctors are able to catch diagnoses earlier or otherwise help improve patient care.

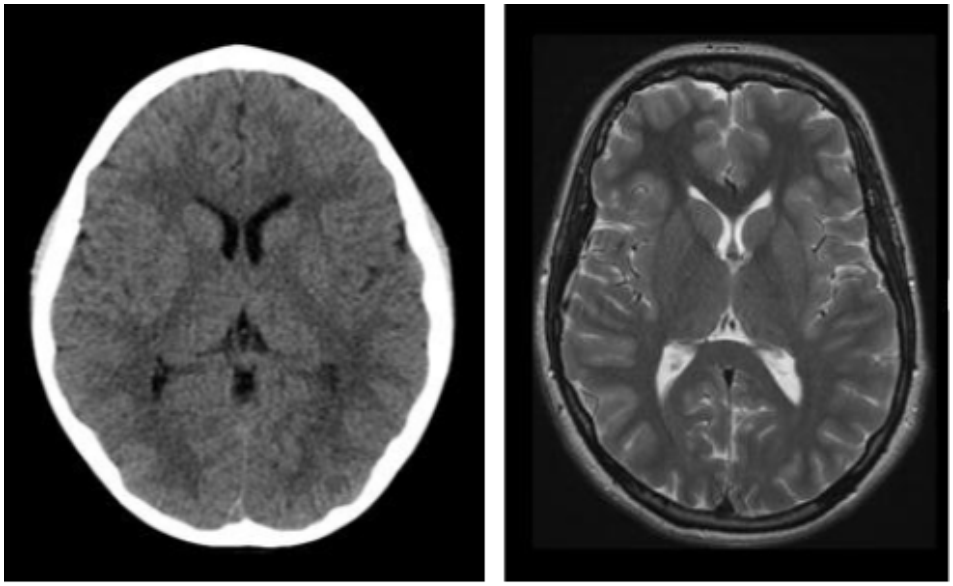


Figure 1.2: Showing the difference between a CT and MRI scan of the brain.

Originally these scans were printed on film and viewed on a light-box. Radiologists, who are responsible for interpreting and sometimes acquiring these images, often viewed these films in a dark room to help make sure that the viewing conditions were as good as possible. The viewing conditions could either enhance or degrade the subtle details of the film. Starting in the mid 1980's these scans started to be digitized and viewed on monitors. Initially this was problematic because most monitors are only capable of displaying images with 8 bits per sample. However these scans are grayscale images, generally with 10-12 bits per sample of data. This lead to the development of special high resolution monitors that display luminosity only - meaning they are capable of displaying those 10-12 bit grayscale images. An example of this display system can be seen in Figure 1.3.

Initially the major limitation for these displays was the processing power needed to update and render the images to the screen fast enough to provide reasonable interaction for radiologists. Current processing power has helped to overcome this limitation - so much so that there has been a more recent movement towards moving past 2D displays into 3D displays. Since CT's

Figure 1.3: PACS display system from the late 1990's [16]

and MRI's consist of multiple slices of the body, it is a natural extension and arguably a more helpful way to display that 3D data as a volume or in a Virtual Reality (VR) environment. A major drawback from stepping away from those grayscale HDR monitors, however, is that VR headsets can only display 8 bits per sample. So when moving to a Virtual Reality environment, it is important to make sure that as little information as possible is lost when converting from the 10-12 bits of data of the initial scan to the 8 bits available in the VR display.

1.4 Contributions

This thesis offers three novel contributions. First it introduces a real-time implementation of 3D Contrast Limited Adaptive Histogram Equalization. This allows medical professionals to interactively enhance the contrast of a medial volume or image stack. It also introduces two new extensions to the real-time algorithm, which give the user more flexibility and control over

the contrast enhancement. The first extension is a focused version, which applies the contrast enhancement algorithm to a specified block of the medical volume. The second extension is a masked version which applies the algorithm to a particular organ or organs.

Chapter 2

HDR Images

2.1 Capturing HDR Images

The range of colors that we are able to perceive with our eyes is often not possible to capture with the limited brightness or luminosity values of Standard Dynamic Range Images. When trying to capture images, containing values outside the Standard Dynamic Range (SDR), there will be areas within the image that are either over or under exposed.

Figure 2.1 shows a range of SDR images taken at increasing levels of exposure from left to right. The leftmost image was taken with the shortest exposure time. As a result, only the details in the foreground are visible. Everything in the background beyond the window is underexposed and too dark because of that short exposure time. As the exposure time is increased, more and more of those background details become visible; however, the foreground details get over exposed or washed out and loose their detail. The right most image was taken with the longest exposure time and is practically the inverse of the first image. It is able to capture all the detail in the background, yet none in the foreground. Professional photographers can play with the exposure time to capture the detail as best they can; however, they will not be able to fully capture the details in both the light and dark regions of these images without increasing the

Figure 2.1: Range of Standard Dynamic Range Images taken at increasing exposure times from left to right.

dynamic range.

Since there is not one single exposure time that is capable of capturing detail throughout the entire SDR image, High Dynamic Range imaging is needed in order to capture all the details in the scene. Multiple methods have been presented to combine a series of SDR images, as seen in Figure 2.1, to create a single HDR image. In general, the methods work by first estimating the camera response function. This function is the mapping from the colors we see with our eyes (also referred to luminance or radiance) to the numerical values stored in the SDR image. Once an estimation for this mapping is calculated, the original SDR images can be combined together to create a single HDR image by taking a weighted average of the SDR images. The weights are based on the mapping and the known exposure times or ratios [4, 18]. The final result using the algorithm presented in [18] can be seen in Figure 2.2. Notice that the details in the foreground as well as the background behind the window are visible, whereas this was not possible with any of the SDR images from Figure 2.1.

Since these methods require multiple images to be taken at different exposures, the best results occur when the captured scene is static. Dynamic scenes would have inconsistencies between the different images and produce blurring or ghosting artifacts in the final HDR image. This restriction to static scenes motivated the development of other HDR imaging methods that only use one image. Only requiring one input image allows dynamic scenes to be captured in HDR. These methods are possible through improvements or changes to the camera's imaging

HDR Image | Low Exposure SDR Image | High Exposure SDR Image

Figure 2.2: Resulting HDR image when applying the algorithm presented in [18] to the 5 images seen in Figure 2.1. Reproducing one of the low and high exposure images for comparison.

sensors. One such approach is to spatially vary the pixel sensitivities of the imaging sensor. This is accomplished with masks of different optical transparencies next to the detector sensor of the camera. This causes the pixel values of an image to have different exposures for neighboring pixels, which is how this method is able to capture information from multiple exposures in a single image. These resulting pixel values at varying exposures are then scaled by the exposure to generate the appropriate radiance values. The final HDR image is then reconstructed by interpolating between these radiance values [21]. Results from this method can be seen in Figure 2.3. The left image is the acquired SDR image, with the zoomed in portion showing the varied exposures of those pixel values, and the right image is the resulting HDR image.

Since HDR images contain data that ranges beyond the Standard Dynamic Range, the methods to create and capture them require more information than is captured in a single SDR image. Different methods gather this information in different ways. Some utilize multiple images taken over varying exposures, whereas others use just one image taken with multiple or specialized camera sensors. Regardless of the method, gaining this data is a necessary step in the process of creating High Dynamic Range images.

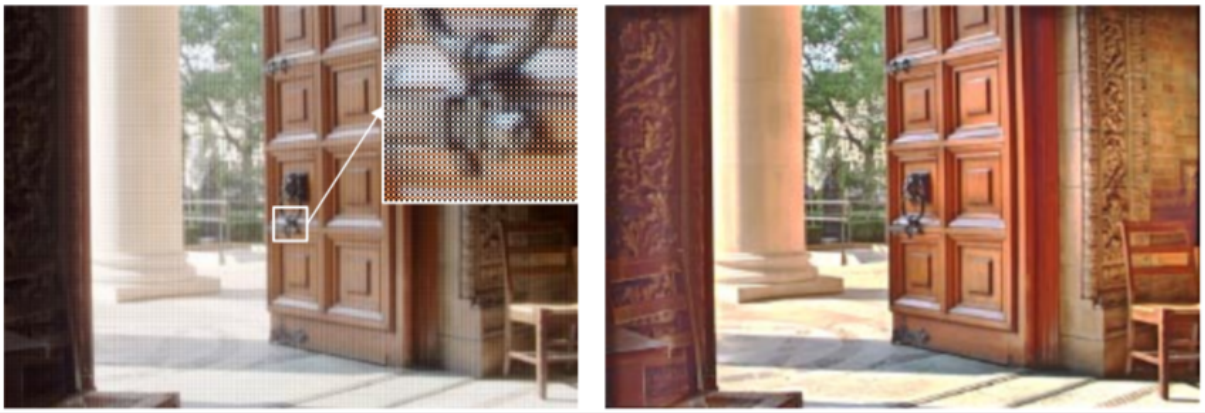


Figure 2.3: Figure from [21] showing the acquired image on the left, zooming in to see the varied exposures of the neighboring pixels, and the resulting HDR image on the right.

2.2 Displaying HDR Images

High Dynamic Range images contain more information and detail than Standard Dynamic Range images. To truly display all that detail, an HDR monitor or display is needed. However, most monitors are only capable of displaying SDR or 8 bit data. Here lies the main disadvantage of HDR images. Simply having an HDR image is not enough. In order to see all that data accurately, an HDR display is also needed. While HDR images exist, they are uncommon. This has sparked research into developing methods for compressing HDR images down to 8 bits so that they can be displayed on the common SDR 8 bit display.

A common technique is called tone mapping. In general, tone mapping functions map colors from some initial dynamic range into another dynamic range. How this mapping is done varies depending on the desired result of the image. Some tone mapping methods aim to bring out the details in an image, increase the contrast, or accomplish a particular look or effect. Tone mapping functions can either be global of local functions. Global methods apply the same function to the entire image, whereas local methods alter each pixel value based on its surrounding features.

The simplest method to map an HDR image to an 8 bit SDR image is a global tone mapping function defined by Equation 2.1. Where V_{HDR} is the HDR pixel value, V_{max} is the

largest HDR pixel value ($V_{max} = 2^{16} - 1 = 65535$ for a 16 bit HDR image) and V_{SDR} is the resulting SDR pixel value. Since it is a global method, this equation would be applied to each pixel in the original image to get the corresponding SDR pixel value. The final SDR image would then approximate the appearance of the original HDR image. While it is not possible to exactly represent a 16 bit HDR image in SDR, its appearance can be approximated. These approximations are improved by retaining localize contrast, preserving image details and color appearance.

$$V_{SDR} = \frac{255}{V_{max}} \cdot V_{HDR} \tag{2.1}$$

Another common method for HDR compression is called gamma correction. It can be used in combination with or after applying a tone mapping function. It follows the formula seen in Equation 2.2. The gamma value can adjust the contrast of the image. Decreasing γ will increase the exposure of the underexposed parts of the image and result in a lower contrast image overall, whereas having $C < 1$ can decrease the exposure of the overexposed parts of the image. There are some specific values for both γ and C that have been developed for the standard color spaces.

$$V_{SDR} = CV_{HDR}^{\gamma} \quad C > 0, \text{ and } 0 < \gamma < 1 \tag{2.2}$$

Ideally, when compressing HDR images into the Standard Dynamic Range they appear as similar as possible to the original HDR image. The main factor that contributes to this similarity is the intensity ratio or image contrast between the original HDR and SDR images. It is only possible to preserve the original image contrast on a device if the dynamic range of the device matches or exceeds that of the image. This means that it is impossible to exactly maintain the image contrast of an HDR image on a SDR display due to the differences in dynamic ranges. Instead, these compression methods aim to minimize the perceived differences as best possible. Our eyes have a harder time perceiving contrast differences at higher brightness values than lower brightness values. This means that it is easier to see differences in darker tones than lighter tones.

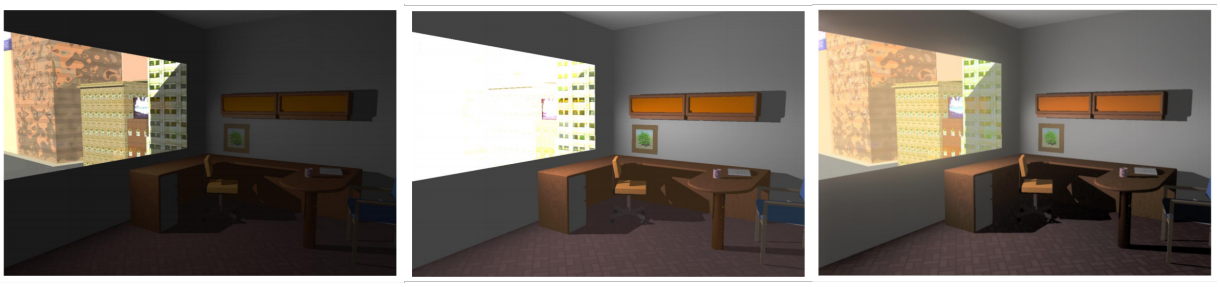


Figure 2.4: Showing HDR images mapped with different Tone mapping functions. The first two either under or over exposes regions of the image, while the third uses spatial information of the image to create the tone mapping function.

Gamma Correction has the ability to smoothly compress the dynamic range of an image; however, it does not preserve the intensity ratios. Bright regions are compressed, while intensity ratios in dark regions are preserved. This is what allows the dynamic range to be compressed while minimizing the perceived differences [5].

All the methods mentioned so far can be placed in the larger classification of Tone Reproduction Curves (TRC). TRC's compress the dynamic range by defining a function that maps the original input intensities into a narrower range of display intensities. These curves are usually image independent; however, there have been methods that take into consideration some spatial properties of the individual image and compress the image data depending on how the local data differs from the dynamic range of the display device [5, 14]. Results from this approach can be seen in Figure 2.4. This figure shows the results of three different tone mapping functions on the same synthetic image. The first underexposes the indoor region, while the second overexposes the outdoor region. The method presented in [14] is able to avoid under or over exposing regions of the image because it also considers spatial properties of the image itself, where as the first two tone mapping functions do not.

Another approach to compressing High Dynamic Range images for SDR displays is called the Multi-resolution algorithm. This works by decomposing the original HDR image into multiple images of different spatial resolutions, then compressing each of these sub images individually before reconstructing the final image. Decomposing the image is done by applying multiple

Figure 2.5: Results from [7]. The right images show the decomposition of the original HDR image. From top to bottom they are the Base, and Detail layers as well as the Color. The left shows the final image.

low-pass filters to the original image. These resulting images are typically compressed using TRC's like the gamma correction function. The main disadvantage to this approach is that edges in the original image can become distorted, and produce unwanted artifacts in the final image like banding or halos [5]. A similar approach that aims to remove these artifacts is called Fast Bilateral Filtering. The original HDR image is decomposed into two different images - a base layer that encodes the large-scale variations, and a detail layer that encodes the fine grain details. The base layer has its contrast reduced with the bilateral filter, and is combined with the detail layer whose contrast was not reduced. By not compressing the detail layer, more of the original HDR detail is preserved in the final image [7]. The results from this method can be seen in Figure 2.5. The main image on the left is the resulting SDR image. The top right and middle images are the decomposed base and detail layers.

All the mentioned approaches are attempting to take the information in HDR images and efficiently map them into the Standard Dynamic Range. They are trying to improve the SDR approximation of HDR images by retaining the local contrast while preserving the image details and color appearance.

Chapter 3

Medical Imaging

3.1 Medical Images

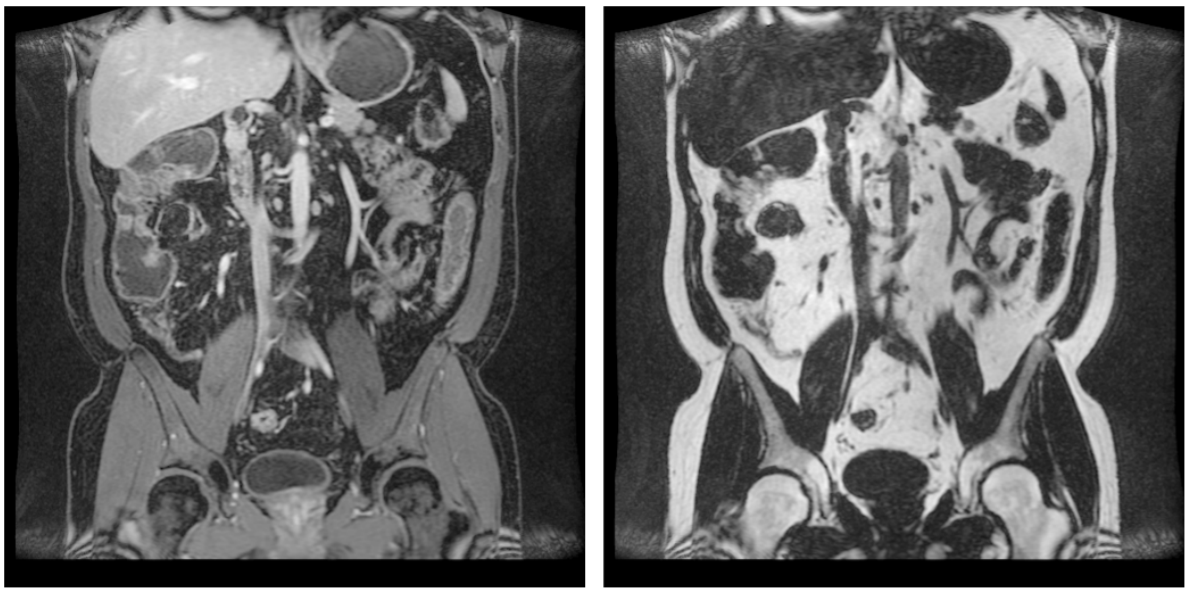
Medical images have been traditionally captured on film. It was not until the 1980's that those films were transferred into digital images. With this change came the need to easily share and view the scans, which brought about two developments PACS and DICOM. PACS or Picture Archiving and Communication System, is a system which provides storage and access to images through various devices. PACS is a unifying system for medical imaging and was integral to the transition from hard-copies of scans to digital copies or soft-copies. As a part of that transition, PACS assisted the development of devices to view the digital scans within the hospitals, as well as provided ways to view medical images off-site and integrate them within existing patient history and radiologist workflows. The data format used within PACS for image storage and transfer is called DICOM. DICOM stands for Digital Imaging and Communications in Medicine and is the standard for the communication and management of medical imaging information and data. DICOM files have specific formats and network protocols that help to easily share and view medical images. Both PACS and DICOM are used across the world and have been central to the development of digital medical imaging.

DICOM files incorporate the standards set for almost every type of medical imaging. In addition to the scan data, DICOM files store patient and doctor information, as well as details specific to the scans within their metadata. For MRI scans, the actual scan data is stored as a series of 12 bit grayscale images with each slice of the scan stored as a separate image. MRI's work by exciting the protons in the body with pulses of radio waves, and use external magnetic fields to detect the affect of those pulses. Since different tissues are more responsive to different types of radio waves, varying the parameters of the pulses varies the resulting contrast generated and the types of tissue that is detected in the scan. For example, some sets of parameters cause fatty tissues to show up brighter in scans, whereas others can enhance watery tissues. Two such parameter settings are highlighted in Figure 3.1. It shows two MRI scans for the same patient, taken with different parameter settings. The left scan used parameters that enhanced the watery tissues, whereas the right scan used parameters that increased the visibility of fatty tissues. Looking at the middle sections of these scans, the organ and vein details are much clearer in the left scan since they are both composed of watery tissues. Whereas those details are lost in the right scan.

3.2 Medical Imaging Displays

PACS systems incorporate display devices and systems, and have come a long way since its inception. The first types of displays were very limited in their processing power, which meant that the ability of the doctors to interact with the scanned images was very limited. As a result, research was focused on optimizing their performance to improve the image interaction [16]. An example of this type of display system can be seen in the right image of Figure 3.2.

These first displays were mainly helpful in determining what aspects were important in an ideal display. Part of that ideal display was the ability to represent as much of the detail as possible that was in the original film scans. This lead to research into the human visual system



WaterPoseCorLavaFlex2 FatPoseCorLavaFlex2

Figure 3.1: MRI scans using two different sets of parameters allowing different types of tissue to be highlighted in the final scan.

and what gray values the human eye could detect. It was concluded that the human eye could detect around 720 different shades of gray. Since 9 bits of gray levels corresponds to 512 shades of gray, and 10 bits corresponds to 1024 shades of gray, having a 10 bit display would be surpass the amount of gray values human eyes can distinguish [13]. In the late 1990's and early 2000's the Grayscale Standard Display Function was developed. This standard was based on the research into the human visual system, and was then used as a target for all display systems to help improve the display the digital scans. These 10 bit HDR grayscale monitors helped to view the detail from the original scans. These monitors, in combination with 10 bit gamma correction tables, allowed for different scans to appear the same even on different displays [16]. These hardware advances overcame the underlying problem of how to display the HDR digital scans. Figure 3.2 shows an example of these displays in the left image.

As mentioned, DICOM is the standard format for storing medical images, so naturally numerous DICOM viewers have been developed that allow a user to view DICOM files on their available display. Most of these are 2D viewers, which means that a user can view an MRI scan

PACS workstation 1980's - single display unit running on a UNIX workstation

PACS workstation early 2000's running 2MegaPixel quad displays

Figure 3.2: Different PACS workstations and displays. The earliest phase of workstations is on the left, with a more advanced display system on the right. [16]

one slice at a time. The interaction available generally includes zooming in and out of the image, the ability to take measurements, and perhaps apply some preset filters to the image to increase or decrease the contrast. There are some 3D DICOM viewers that create a 3D visual representation of the 2D slices that can then be viewed on the available 2D display. In general the interaction with these viewers are the same as their 2D counterparts. There have been some more recent developments using Virtual and Augmented Reality to display and interact with 3D scans in an immersive environment. However, they are still in development and suffer from that initial shortcoming of the first PACS display systems. They are all limited to displaying within the 8 bit Standard Dynamic Range. While the HDR grayscale displays found a hardware solution to display the details in medical scans, the same solution is not currently available in Augmented Reality and Virtual Reality headsets. Overcoming this mismatch, and helping to improve the 3D visualization of these Medical Images in real-time is the focus of my research.

Chapter 4

Previous Image Enhancement Techniques

4.1 Histogram Equalization

Image Processing techniques have been around for decades. One method that has sparked a lot of research is Histogram Equalization [9]. This method was developed in the 1970s and has the goal of increasing the global contrast of an image. This method is especially useful when applied to images that are either over or under exposed. This method has also been shown to produce particularly good results in x-ray images [6]. Histogram Equalization works by redistributing pixel values for an image throughout the entire dynamic range. Take for example the image shown on the top left of Figure 4.1. The majority of the pixel values are concentrated between 125 and 200. Using Histogram Equalization on this image will spread out that range to cover the entire dynamic range, which in turn increases the global contrast of the image.

The process of applying Histogram Equalization to an image starts by creating a histogram, which means counting the number of times each gray values within the dynamic range occurs in the image. The histogram for the example image is in the top right of Figure 4.1. Once the Histogram has been calculated, the normalized Cumulative Distribution function (CDF) is calculated to create a mapping between the current pixel value to the new equalized gray value.

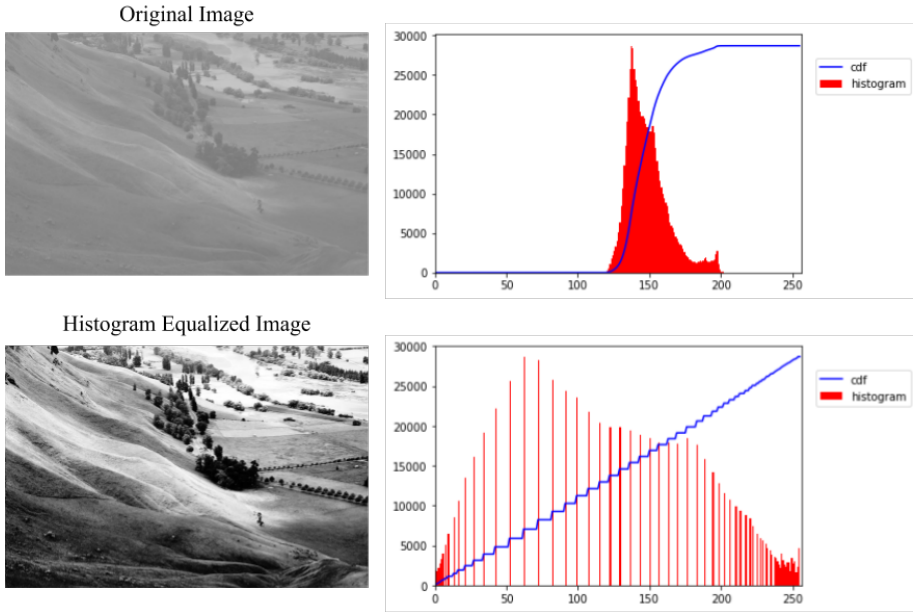


Figure 4.1: Before and After applying the Histogram Equalization method to a 256 jpg image

This mapping is what redistributes the pixels so that they are more evenly spaced throughout the entire dynamic range. Once this mapping is applied, those new pixel values will span the entire dynamic range and visually results in enhanced global contrast of the image. The bottom image in Figure 4.1 shows the resulting Histogram Equalized image, and its corresponding histogram.

This method is fairly simple and straight forward, and despite being computationally inexpensive, it is still able to improve the global contrast of images. Its main disadvantage however, is that this contrast enhancement is applied indiscriminately. If the original image has a fair amount of noise in it, applying this technique will not only enhance the contrast of the overall image, it will also enhance the noise. This makes any noise in the original image much more visible in the Histogram Equalized image.

4.2 Local Histogram Equalization

Histogram Equalization enhances the contrast based on the pixel distribution of the entire image. As seen in Figure 4.1, while the overall contrast was significantly improved, there are

still areas of the image that appear over or underexposed. The cluster of trees in the center of the image are underexposed, where as the river in the top right of the image is overexposed. Local Histogram Equalization or Adaptive Histogram Equalization was developed to help combat this effect and improve the contrast of these local regions.

The idea of Adaptive Histogram Equalization (AHE) is to create local histograms for each of the pixels in an image, so that the final enhancement adapts to these smaller images sections. Initially Adaptive Histogram Equalization, used a sliding window approach. This approach creates and computes a local histogram and gray value mapping for each individual pixel using its surrounding pixels. Unexpectedly, using such specialized histograms is very computationally expensive [15]. To help speed up these AHE methods, the idea of neighborhoods was proposed. Neighborhood methods use varying metrics to determine the pixels that should be included in the local histogram for each of the pixels. [8, 23]. Other methods use a global Histogram Equalization approach that takes into consideration additional local information to avoid per-pixel histograms [3]. Others have more specialized cumulative distribution functions that can be modified to adjust the final image results [30].

Another method aimed at reducing the computational complexity recommended breaking apart the image into different sections, called Contextual Regions (CR), and then applying the Histogram Equalization technique to each of those sections individually. It then bi-linearly interpolates between these sections to create the final image [25]. This allows the calculated Histograms and Cumulative Distribution Functions to have mappings that are more optimized for a particular region of the full image, without the expense of creating a different local histogram and mapping for each individual pixel. This adaptive version is able to enhance the image in a more specialized and adaptive way. Meaning that the contrast of an overexposed region in an image will be improved and no longer appear as overexposed.

Adaptive Histogram Equalization does a better job of enhancing the image and bringing out more details in the different sections of the image than the original Histogram Equalization

Original Image | Histogram Equalization | Adaptive HE (4x4 CR)

Figure 4.2: Comparison between the original Image, and the enhanced image using Histogram Equalization, and Adaptive Histogram Equalization with 4x4 Contextual Regions.

technique. It improves the contrast of those previously overexposed and underexposed regions. Figure 4.2 shows a comparison between the Adaptive and original Histogram Equalization methods. Areas that the original Histogram Equalization method left over or underexposed are no longer over or underexposed with the Adaptive method, and more detail is visible with Adaptive method. For example, the river in the top right of the AHE image in Figure 4.2 is no longer overexposed, and the details are much clearer. Overall this method does a much better job at enhancing the smaller details within an image.

Since Adaptive Histogram Equalization breaks apart the image into smaller sections or contextual regions, varying the number of contextual regions greatly impacts the final image. The more contextual regions used, the smaller they become, which results in histograms and mappings that are more specialized to a particular region. The more specialized these mappings become, the more detail that can be extracted from that section of the image. Figure 4.3 shows the results from varying numbers of contextual regions. The river in the upper right of the image, shows how much detail can be enhanced with the smaller contextual regions. Similarly, contrast on the trees in the center of the image allow us to see much more detail than we otherwise would be able to see.

An important aspect to note is that as with Histogram Equalization, Adaptive Histogram Equalization is also susceptible to enhancing noise. If the number of Contextual Regions is large enough that the resulting area inside a single Contextual Region is fairly uniform in color,

Figure 4.3: Adaptive Histogram Equalization with varying numbers of Contextual Regions.

applying Histogram Equalization will map that narrow range of values to the entire dynamic range. So any slight variation or noise in the original image will be amplified in the final image.

4.3 Contrast Limiting

As mentioned, a drawback to both Histogram Equalization and Adaptive Histogram Equalization is the possibility of magnifying noise within the original image. To counteract this problem some methods were developed to use a combination of both global and local histograms to help reduce the additional noise [34]. Another method is Contrast Limited Histogram Equalization (CLHE). CLHE aims to limit the noise in the final image by limiting the contrast amplification. In order to do this CLHE clips the histogram at a predefined value, or clip limit, before computing the Cumulative Distribution Function (CDF). This limits the slope of the CDF and therefore the mapping and final contrast of the image [25].

Noise occurs most frequently with uniform images or sections of images. These uniform images result in large peaks in the histogram. Clipping the histogram removes these large peaks and redistributes those values evenly throughout the histogram. This process can reduce the amount of noise seen in the final image. The effect of varying clip limits on the histograms and Cumulative Distribution Functions can be seen in Figure 4.4. Notice that the lower the clip limit, the more uniform the histogram and the flatter the slope of the CDF.

The results of various clip limits with CLHE is shown in Figure 4.5. The higher the clip limit the the more contrast in the final image and the closer the results are to regular Histogram

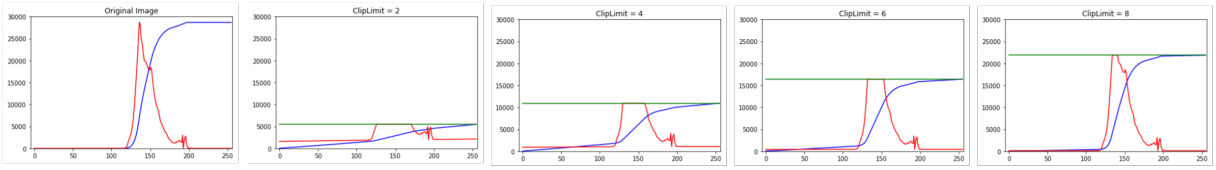


Figure 4.4: Histograms and CDFs before and after applying CLHE with varying values for the clip limit.

Equalization. As that clip limit is decreased, the affects of the Histogram Equalization are minimized, and the less contrast is in the final image. Comparing the original image with the image for CLHE with a clip limit of 2, the enhancement is very subtle. If that clip limit were to go down to 1, it would result in a perfectly uniform histogram and ultimately counteract the contrast enhancement from the Histogram Equalization. In other words using a clip limit of 1 produces the original image.

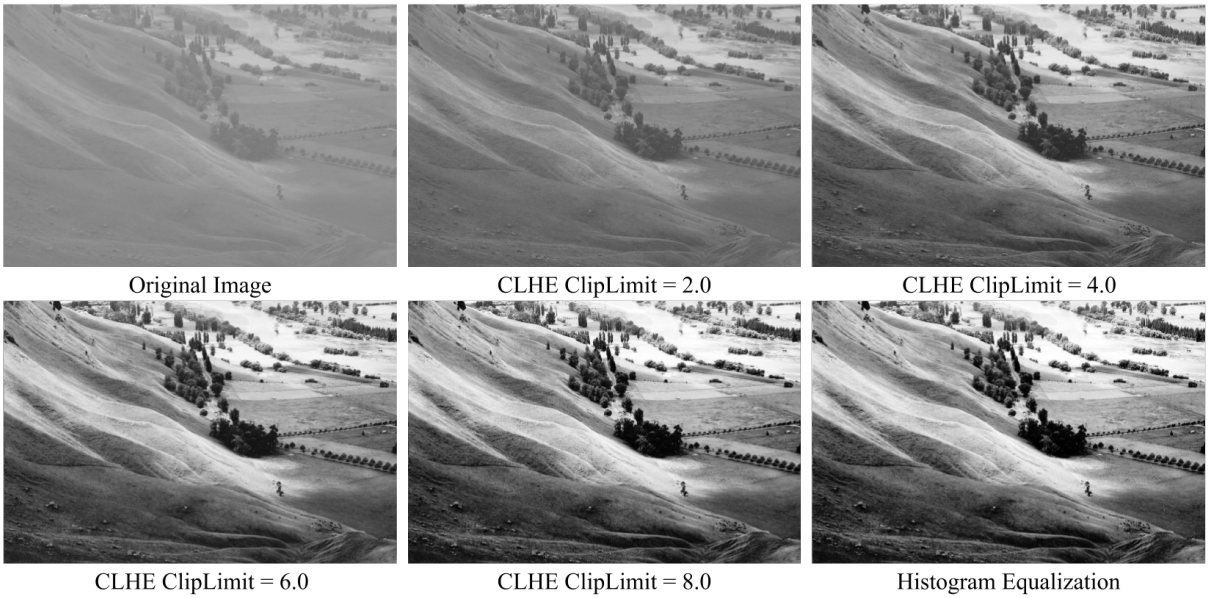


Figure 4.5: Contrast Limited Histogram Equalization applied with varying clip limit Values

CLAHE

The combination of clipping the histograms, along with the Adaptive Histogram Equalization method previously described, creates the method called Contrast Limited Adaptive Histogram

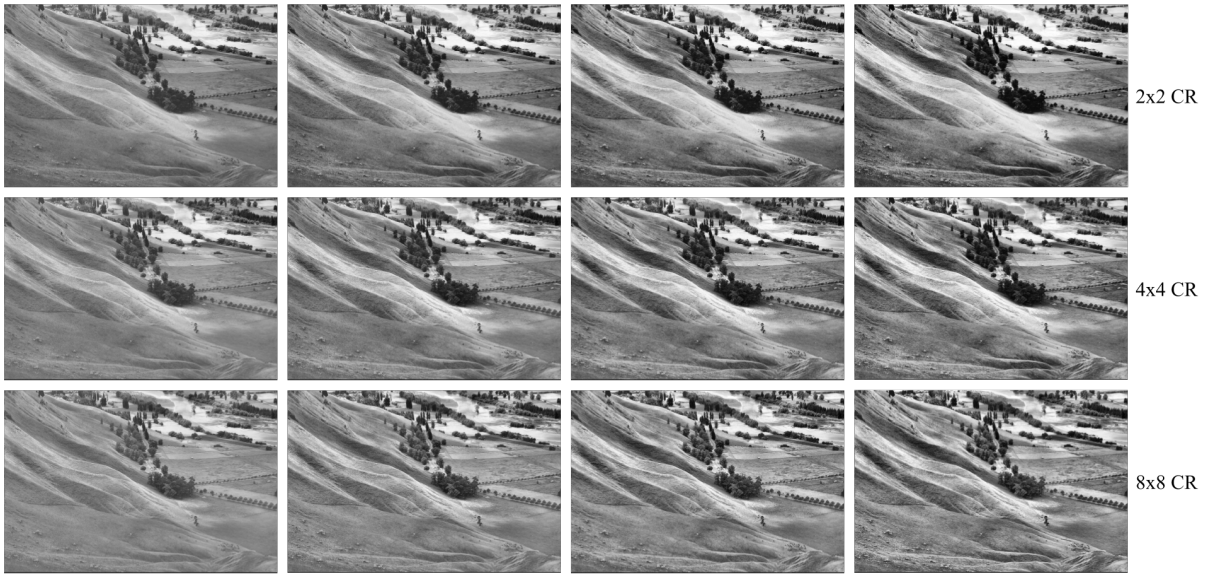


Figure 4.6: Varying Clip Limits and number of Contextual Regions for with the Contrast Limited Adaptive Histogram Equalization method.

Equalization (CLAHE) [35]. This method combines the advantages of the Contrast Limited approach to limit the contrast and noise, as well as the ability to decrease the over and underexposed regions in the final image with Adaptive Histogram Equalization. CLAHE uses the Adaptive method recommended in [25] and divides the image into Contextual Regions (CR) and bi-linearly interpolates between them to generate the final image [35].

This combination of methods provides multiple parameters that can be adjusted for different effects. These parameters allow for flexibility and control over the final enhancement of the image. Adjusting the number of Contextual Regions changes the amount of detail enhanced in the final image whereas the changing the clip limit affects the overall contrast and enhancement of the final image. Results from varying these parameters can bee seen in Figure 4.6. Moving from top to bottom the number of Contextual Regions increases which corresponds to the increase the amount of detail. Moving left to right the clip limit increases which results in a global increase in contrast.

4.4 More HE Techniques

The methods discussed so far have focused on contrast enhancement and noise reduction. Other categories of Histogram Equalization techniques focus on preserving the brightness of the final image or applying Histogram Equalization on color images. Other contrast enhancement methods are done through Fourier transforms and wavelet-based methods, which enhance images through manipulation of the transform coefficients [17].

Preserving Brightness

The methods discussed so far have focused on contrast enhancement and noise reduction. Another category of Histogram Equalization techniques focus on preserving the brightness of the original image. In traditional Histogram Equalization methods the initial brightness of the image is not preserved in the final image. This is clear when looking back at the histograms in Figure 4.1. The average brightness value was originally about 150, however after applying HE the average brightness value appears to have actually decreased. In some cases this can be an undesirable quality. Bi-Histogram Equalization proposed to overcome this brightness preservation problem by separating the original input image into two sub-images based on the average brightness or intensity value [12]. By separating the the image and equalizing the sub-images independently, the average brightness value is maintained after the equalization. There are many different suggestions as to the best way to split the input image into its sub-images. Some techniques recommend the mean, median or some other brightness metric [20].

Another approach focused on improving the background contrast while preserving image brightness by padding the probability density function [28]. Other methods divide the image into four sub-images (Quadrant Dynamic Histogram Equalization) with the advantage that the image is enhanced without any intensity saturation, noise amplification or over-enhancement [22]. Most of these brightness preserving methods were developed to keep the appearance of the enhanced

image similar to the original one. These methods are therefore more commonly used for more consumer electronics where the specific look of the images post enhancement is highly valued.

Color Images

The Histogram Equalization methods discussed so far have all been applied to grayscale images. Applying these techniques to color images requires non-trivial modification. Simply using Histogram Equalization techniques on each color channel individually can cause color imbalance issues. So many different techniques have been developed to help overcome this obstacle. Some of these methods include using a 3D color histogram equalization, instead of the mentioned three 1D histogram equalizations on each color channel. Others spread the histogram along the Principal Component Axis, or apply Histogram Equalization to just the luminance channel of the image [32, 33].

4.5 Medical Imaging

Image enhancement methods vary drastically depending on the type of image and the goal of the enhancement. These are some of the important factors to consider when choosing an enhancement method. Medical images are expensive and it is sometimes difficult for medical professionals to decipher what they need from the images. An area of research has been devoted to developing methods to help improve medical imaging. Since Histogram Equalization is both an effective and simple method for contrast enhancement, it is a popular choice. One of the earlier papers working on medical image enhancement was [24], it focused on the Contrast Limited Adaptive Histogram Equalization approach and argued that it was applicable on a wide variety of medical images. Multiple papers also worked with medical professionals to show that their results and the use of CLAHE was effective and helpful for the doctors diagnosis, and interaction with the medical images [24, 26].

The ability to gather information from medical images is incredibly important. Some researchers have been looking into ways to further improve the visualization of mammogram images to help doctors be able to diagnose patients with breast cancer even earlier. Sajeev [29] applied a Local Contrast Modification. This method is based on the global mean and standard deviation of the entire image, as well as the mean and standard deviation of a local user defined window. After applying Local Contrast Modification (LCM) they apply CLAHE to the image. They found that CLAHE alone tended to over-enhance the image, and lead to the loss of details in mammogram images [19]. The addition of LCM combated this effect and provided a better contrast enhancement while preserving the detail from the input mammogram image.

4.6 3D Medical Imaging

With the development and availability of 3D visualization, multiple papers have taken to adapting 2D methods to 3D. A previous technique of finding a complete ordering of pixels to equally separate the pixels in the equalized image was adapted and optimized for 3D [27]. However, the results were not fast enough for real-time applications.

Since CLAHE is a popular method for medical images, Amorim [2] took CLAHE and extended it for 3D volumes. They presented two different techniques for 3D volumes. The first and simpler approach is to apply CLAHE on each individual layer that makes up the 3D volume. This resulted in inconsistencies between the different layers. Using completely different histograms for different layers meant that after the volume was equalized, neighboring voxels were mapped to completely different values, resulting in those visual inconsistencies. In order to smooth out the equalized volume they proposed the idea of applying CLAHE to a 3D section of the volume instead of a 2D layer of the volume, and tri-linearly interpolating between these resulting histograms. Enhancing the medical data in sub-volumes removed those visual inconsistencies and resulted in a much smoother enhanced volume. Figure 4.7 shows the results of this 3D technique

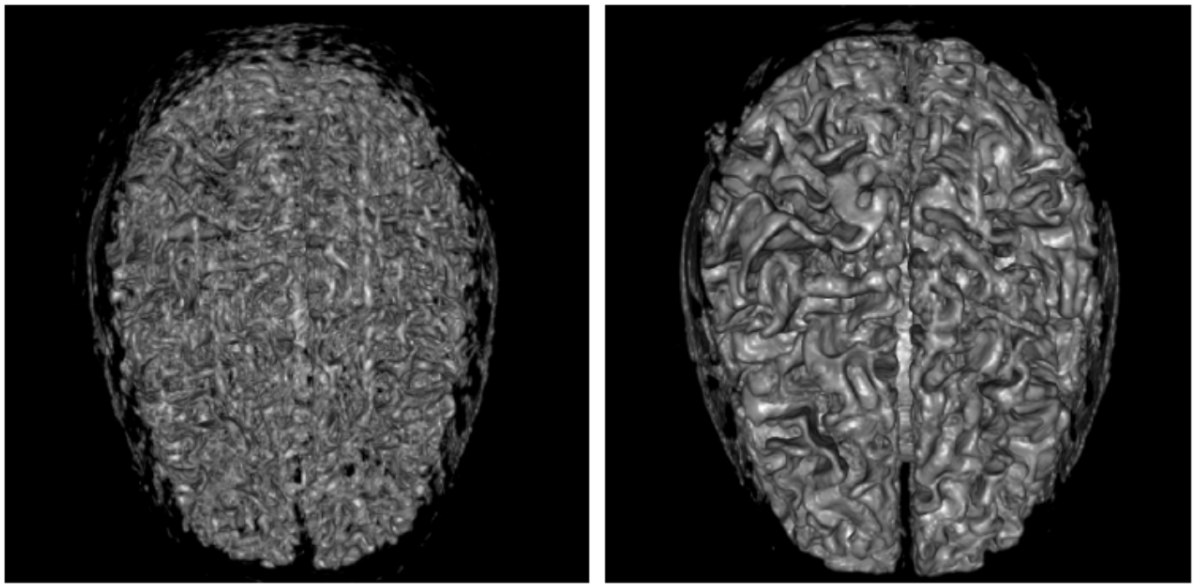


Figure 4.7: Results from method purposed in 3D Adaptive Histogram Equalization method for Medical Volumes [2].

on a brain MRI. 3D CLAHE provided a clear improvement over the 2D method.

It is possible to have a data set with more than three dimensions. For example, a 3D volume has results over time. The previous methods would be insufficient to provide smooth results over that fourth time dimension. In order to accommodate multidimensional data sets, a Multidimensional Contrast Limited Adaptive Histogram Equalization Method was developed to handle any number dimensions. The key component of their approach was their handling of boundaries within the data. In the 2D and 3D versions of Adaptive Histogram Equalization methods, the edges of the images and volumes need to be handled carefully. When generalizing to N-dimensions, these edge cases become increasingly difficult to handle. So they implemented a padding structure to easily generalize the computations for multiple dimensions [31].

Chapter 5

Implementation

5.1 Method

The goal of this thesis is to develop and implement an algorithm that would help improve the contrast of MRI data on an 8 bit display in real-time. So my research was focused on contrast enhancement algorithms, specifically Histogram Equalization methods. Histogram Equalization has been a starting point for a significant amount of research into different techniques for image enhancement. Knowing that I would be focused on enhancing MRI images narrowed down the applicable methods. The method that seemed most appropriate was Contrast Limited Adaptive Histogram Equalization. The Contrast Limited addition to the original Histogram Equalization method was developed to reduce noise specifically for medical images. The Adaptive quality is useful in enhancing details in different sections of the image, which is a valuable quality in an enhancement method. CLAHE overall has a track record of providing good medical imaging enhancement and is a large part of why I chose to focus on CLAHE methods [26, 35].

Since I would be visualizing the MRI data as a volume it was important that the enhancement method also be 3D. One option would be to take the original 2D CLAHE method, or a version of it, and extend it into 3D. In order to accomplish real-time interaction with the algorithm,

it was important to choose an algorithm that would lend it self well to that real-time interaction. One CLAHE method in particular presented really promising results in 2D; however, it worked within a feedback loop, where they need to re-apply their algorithm multiple times in order to create their desired results [29]. This feedback loop would likely be the biggest obstacle for the application to become real-time. The original 2D CLAHE method has been extended into 3D and nD methods [2, 31], however these versions were not built for real-time interaction. I decided to take the 3D CLAHE algorithm presented in [2] and develop it for real-time interaction.

5.2 Initial Implementation

2D CLAHE

I started by developing a Python version of the original 2D CLAHE algorithm [35]. The pseudo code for the 2D version of CLAHE can be seen in Algorithm 1.

Algorithm 1: 2D CLAHE

```
inputs : raw image, number of Contextual Regions, clipLimit
output : Enhanced Image
LUT = createLUT();
foreach Contextual Region do
    buildHist;
    clipHist;
    mapHist;
foreach pixel do
    find 4 neighboring Contextual Regions;
    bi-linearly interpolate between 4 neighbors;
```

The first step in 2D CLAHE is to create a look up table. This look up table (LUT) is used to convert the dynamic range of the input image into the desired output dynamic range. This is done with a simple linear mapping shown in Equation 5.1, where this equation is applied for each of the gray values in the original input image. The $inMin$ and $inMax$ are the maximum and

minimum values of the input image, together making up the input dynamic range. *numBins* refers to the number of gray values within the desired output dynamic range. Meaning that if we want to output an 8-bit image, $numBins = 2^8 = 256$.

$$LUT[inGrayVal] = \left\lfloor \frac{inGrayVal - inMin}{binSize} \right\rfloor$$
$$binSize = \frac{1 + inMax - inMin}{numBins} \tag{5.1}$$

As an example, if we are performing CLAHE on a 12 bit image with the values taking up that entire dynamic range, and outputting an 8 bit image, $inMin = 0$, $inMax = 2^{12} - 1 = 4095$, and $numBins = 256$. This LUT will be the size of the input image dynamic range and will contain the mappings between the original 12-bit gray values to the desired 8 bit output values such that $LUT[inGrayVal] = outGrayVal$. This LUT and the mapping used to generate it is where the dynamic range disparity is handled.

After this LUT is generated, the next step is to create the local histograms. To accomplish the Adaptive aspect of CLAHE, local histograms are needed for each pixel. There are different ways to create these local histograms, but in the interest of efficiency, I implemented the Contextual Region approach mentioned in [25]. This means that for a 2D image, using *nxn* Contextual Regions, I would have an *nxn* array of histograms, where each histogram counts the number of times each of the possible gray values occur within a particular Contextual Region. Once the Histograms have been generated for their respective regions, the histograms are clipped based on a clip value. Any gray values that occur more times than this clip value within a Contextual Region are redistributed evenly throughout that histogram. This redistribution is the Contrast Limited aspect of CLAHE. It is limiting the amount of times a particular gray value can occur in the final image which in turn limits the resulting contrast of the final image. This is done with the goal of reducing the amount of noise in the final image.

The clip limit presented in [35] was calculated as given in Equation 5.2. This one

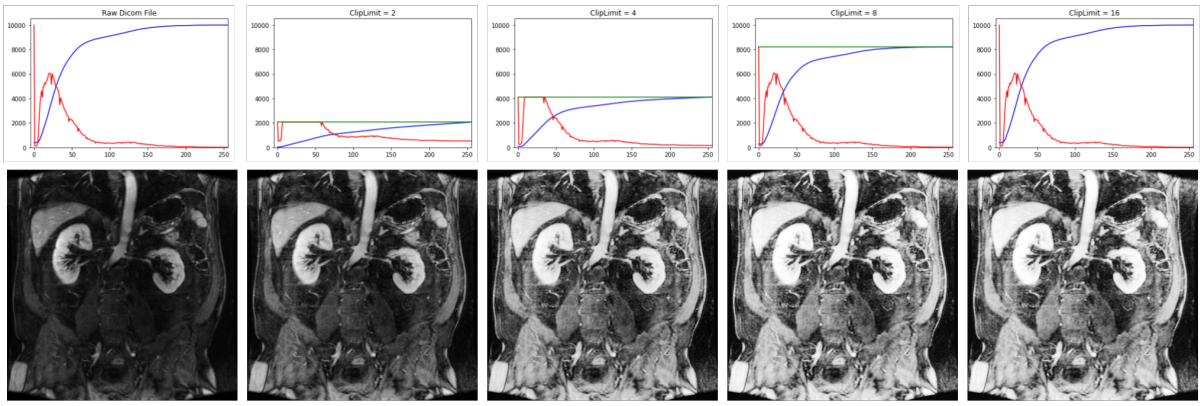


Figure 5.1: Resulting Images as well as plots of the histogram (in red), CDF (in blue), and *clipValue* (in green). Using a single Contextual Region for the entire image and varying values for the *clipValue* calculated with Equation 5.2.

clipValue is used to clip all the histograms such that there is no more than *clipValue* pixels with a particular gray value in any of the Contextual Regions.

$$clipValue = clipLimit \cdot \frac{(sizeCRx \cdot sizeCRy)}{numBins} \quad , \; cliLimit \in [1, \infty) \tag{5.2}$$

The fraction calculates the height of a perfectly uniform histogram, and the *clipLimit* is a user inputted value that can range from $[1, \infty)$. A value of 1 would correspond to a completely uniform distribution of pixels and visually results in an unaltered image. Since this value is used to put a cap on the number of times a particular gray value can occur within a Contextual Region, it is dependent on a variety of factors, including the image size, the number of initial gray values, and the number of Contextual Regions. As a result, there is no consistent value that acts as an upper bound. If the *clipValue* is larger than the count for the most common gray value in a Contextual Region, the histogram will not be clipped and the clip Limit will have no effect on the image. Figure 5.1 shows the histograms and resulting images for the example case of using just one Contextual Region for the image. The red lines are the histogram plots, the blue lines are the Cumulative Distribution Functions for the histogram, and the green line is the *clipValue*. As the clip limit increases, the contrast is also increased; however, the clip limit of 16 results in a

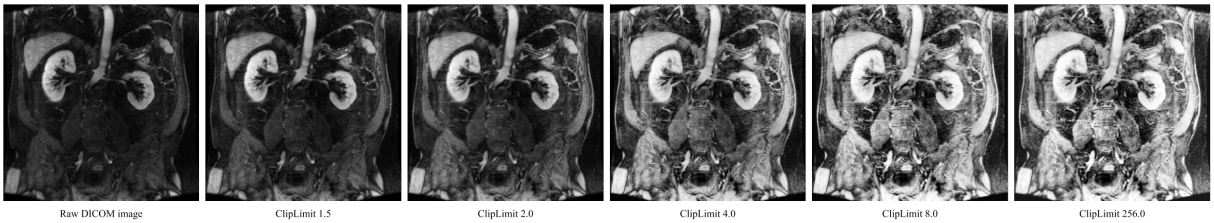


Figure 5.2: Applying CLAHE to a slice of an MRI DICOM with an increasing *clipLimit* where *clipValue* calculated with Equation 5.2.

clipValue that is beyond the most frequently occurring grey value in the image. As a result, the clip Limit or any clip limit greater than 16 will have no effect on the enhancement.

As the clip limit increases, so does the contrast in the final image. Figure 5.2 shows the results of applying CLAHE on a slice of a DICOM file, with 4x4 Contextual Regions and varying clip limits. Again notice that after a point, the increased clip limit no longer affects the final result because the corresponding clip value is larger than the maximum value in any of the histograms. The value for the clip limit can be thought of as how far away the histograms can stray from a uniform distribution. Through experimentation, values between 2 and 8 seem to give a good range of results, but the results do vary between different amounts of Contextual Regions and different sized images.

This clip limit is constant throughout the adaptive equalization process. Meaning that the same *clipValue* is used for every local histogram. So it is possible that the clip value only clips some of the histograms. I implemented a more adaptive approach to the clip limit in which it is treated as a percentage of the most frequently occurring value in a particular Contextual Region (Equation 5.3. With this approach the *clipLimit* ranges between $[0, 1]$, with a value of 1 corresponding to the fully enhanced contrast, and the smaller the value corresponding to less contrast in the final image. It is important to place a lower bound on the *clipValue* to make sure that it does not go below what a uniform distribution would be. It would be impossible to re-distribute the values of the histogram if they were clipped below this uniform distribution value. To find the *minClipVal* or lower bound, I utilized the original formula for the *clipValue* with 1.1

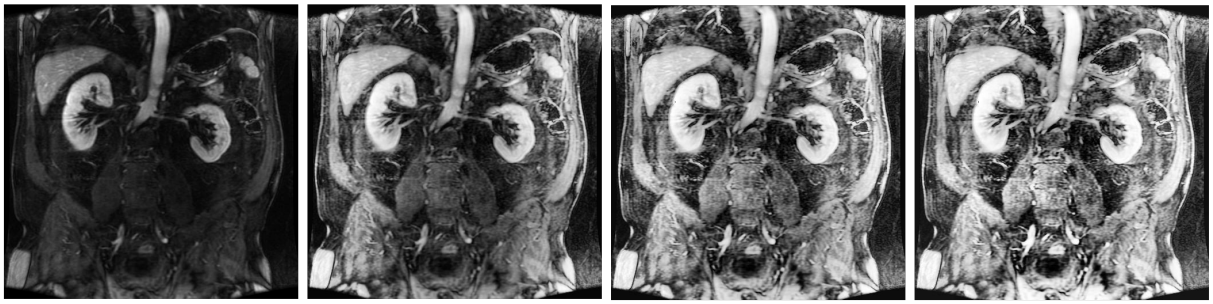


Figure 5.3: Applying CLAHE to a slice of an MRI DICOM with an increasing *clipLimit* where *clipValue* is calculated with Equation 5.3.

as the clip limit. I chose 1.1 because the closer the *clipValue* is to a uniform distribution, the harder it gets to re-distribute the histogram values. Figure 5.3 shows the results of this approach for varying clip values on a DICOM slice, again using 4x4 Contextual Regions.

$$\begin{aligned} clipValue &= max\big(minClipVal,\ clipLimit \cdot max(hist_{currCR})\big) \quad ,\ cliLimit \in [0,1] \\ minClipVal &= 1.1 \cdot \frac{(sizeCRx \cdot sizeCRy)}{numBins} \end{aligned} \tag{5.3}$$

The difference between these two approaches is subtle. The original approach treats the clip value as a global clip on the histograms, which results in a smooth transition between the raw image and the CLAHE enhanced image. On the other hand, the percentage approach depends on the local histograms, which results in a local *clipValue* that adjusts the contrast of each local region individually. Figure 5.4 shows the comparison between these two approaches using 4x4 Contextual Regions on a single DICOM slice. To make the comparisons as close and fair as possible, the clip limit used in the original global method is equivalent to the average clip limit applied in the local approach. So the overall contrast is about the same for these sets of images; however, differences can be seen between the two sets of images. The original method seems to over expose the center bottom of the images especially in middle and right images. Whereas the details in this region are better preserved in the local approach. The rest of the results presented

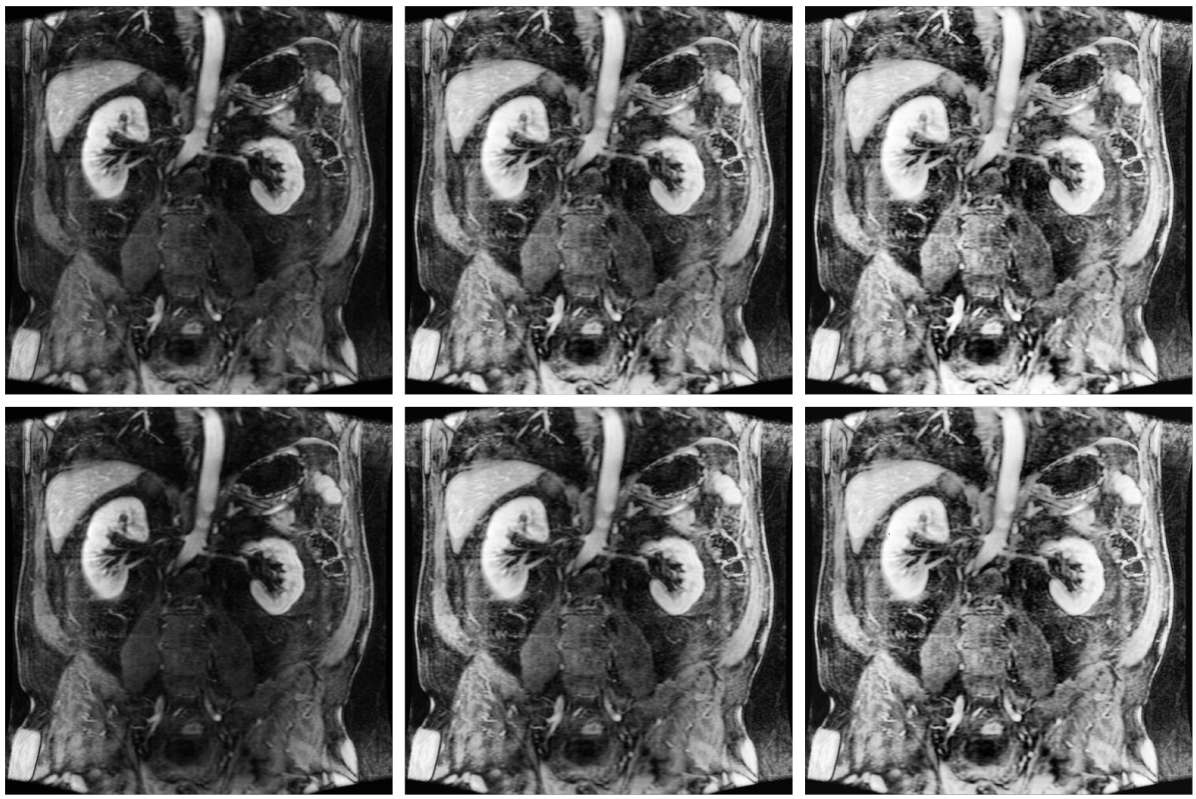


Figure 5.4: Comparison between using the *clipValue* as calculated in the original CLAHE paper (Equation 5.2, with the presented adaptive/local approach (Equation 5.3).

will be using this local approach to the clip limit.

Once the histograms have been clipped, the cumulative distribution function is calculated for each histogram to create the mapping between the original gray value and the CLAHE enhanced gray value. These mappings will vary between the different Contextual Regions, which means that if they were applied only to the pixels within their Contextual Regions, there would be very distinct lines in the final image. In order to create a smooth final image, the neighboring Contextual Regions need to be found for each pixel, and final pixel value is determined by bi-linearly interpolating between those different Contextual Region mappings.

3D CLAHE

While 2D CLAHE works well for images, running the 2D algorithm on each slice of a volume is not practical and does not result in a smooth volume. So I followed Amorim [2] to implement 3D CLAHE.

Since the majority of the operations are done within the histograms, the transition from the original 2D CLAHE to a 3D CLAHE is fairly straightforward. In the 2D case, the image or slice was broken up into a 2D grid of Contextual Regions, so the 3D case needs to break up the volume into a 3D grid of Sub-Blocks. So instead of accumulating the pixel values over the Contextual Regions, it needs to be done over these Sub-Blocks. Calculating the LUT as well as clipping the histograms and re-distributing those excess pixel values remains the same. The only portions of the algorithm that needs to accommodate the 3D change is the initial histogram calculation and the final interpolation. In the 2D case, the 4 neighboring Contextual regions were found for each pixel. These are the Contextual Regions that are above and below the pixel in both the x and y directions. Extending to 3D means that there are 8 neighboring Sub-Blocks that surround the pixel in all three dimensions. The results of the 3D implementation of CLAHE can be seen in Figure 5.5.

5.3 Optimizing for Real-Time Interaction

The initial implementation was done in Python because of its simplicity when working with images. However, it became clear that the speed limitations of the language would be unacceptable for a real-time application. Running the algorithm on just a DICOM slice would take about 2.2 seconds, and the entire volume took about 5 minutes. These are nowhere near acceptable run-times for a real-time application. The fist change to speed up the computation was to re-implement the algorithms in C++. This version followed the exact same implementation as the Python version, and even without the use of any other optimizations such as threading,

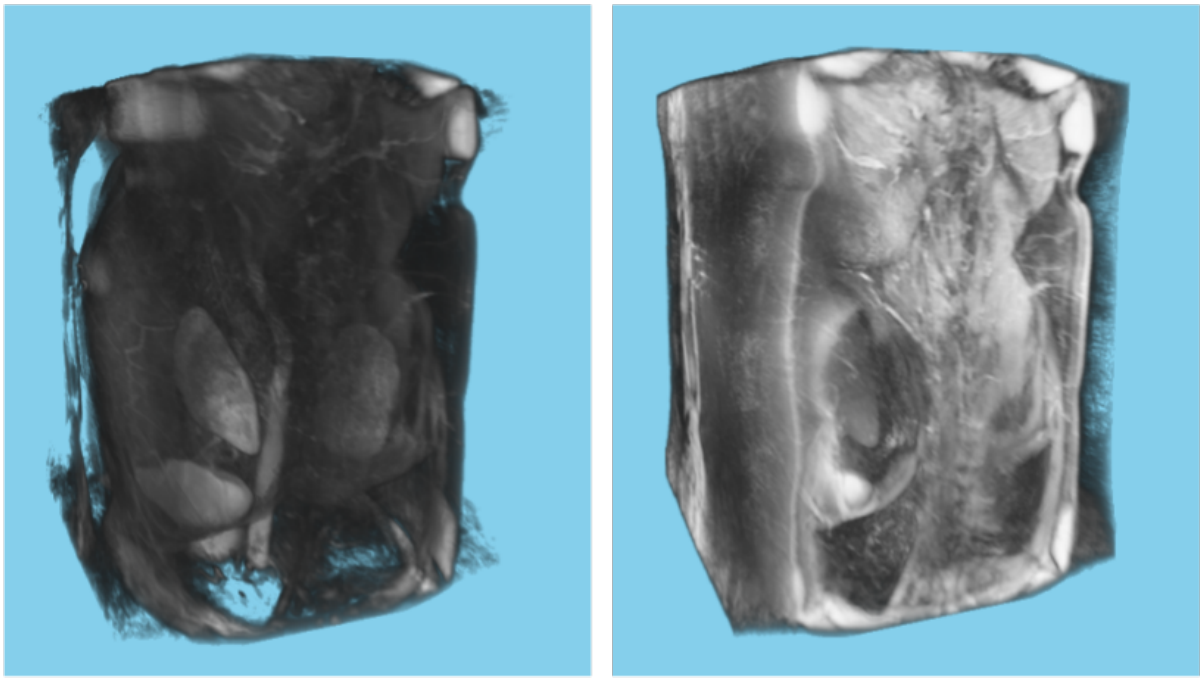


Figure 5.5: Comparison between the original DICOM volume on the left, alongside the CLAHE enhanced volume on the right.

this change alone produced a considerable improvement. The speed up for computing the entire volume went from 5 minutes to 3 seconds. Despite the significant increase in speed this is still slower than would be preferred for real-time. To further optimize, the algorithm was transferred onto the GPU using GLSL Compute Shaders.

GLSL Shader Background

OpenGL or Open Graphics Library is a graphics platform designed for rendering 2D and 3D graphics. The OpenGL Pipeline starts with the 3D data of a scene and transforms that data into the 2D pixel data that is displayed on the screen. An outline of this pipeline can be seen in Figure 5.6. Parts of this pipeline are taken care of by OpenGL, however there are a couple places the programmer can change. These changes are implemented through what are called Shaders. Shaders are small programs written in GLSL (OpenGL Shading Language), which is a C-like language that is tailored for use in graphics. It has many built in common variables and functions

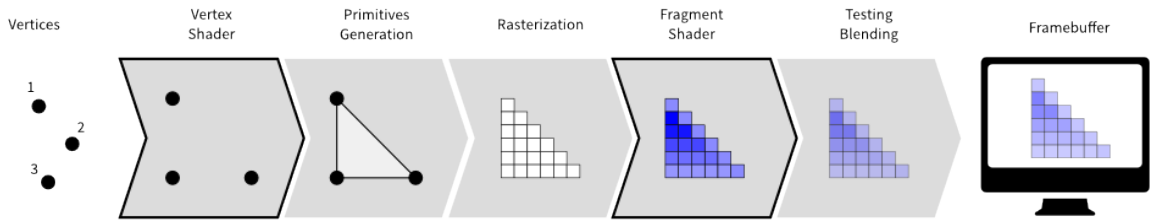


Figure 5.6: Outline of the OpenGL Pipeline, the Vertex and Fragment Shaders are portions that can be written to alter what eventually is displayed on the screen.

such as matrix and vector operations. In OpenGL, there are three common types of shaders; Vertex Shaders, Fragment or Pixel Shaders, and Compute Shaders. Vertex and Fragment Shaders are used in a specific order within the OpenGL pipeline. The Vertex Shader is the first part of the pipeline and processes those initial 3D coordinates. The output of the Vertex shader is passed along the pipeline and is eventually the input to the Fragment Shader which determines the color of each fragment or pixel on the screen. The operations of these shaders are very specific and easily executed in parallel, because of this each of these programs are executed on the GPU.

Vertex and Fragment Shaders live in a particular place within the pipeline with a well-defined set of input and output values. The Vertex Shader is executed once per vertex, and the fragment is executed once per fragment. Compute Shaders on the other hand do not live within this pipeline, and do not have a predefined set of inputs and outputs. This gives a lot of flexibility in how they can be used. In general they are used for General Purpose GPU (GPGPU) to compute arbitrary information on the GPU. The flexibility of Compute Shaders means, that in addition to the program details for the actual computation, the user needs to specify how many times to execute the program, as well as specifying the inputs and outputs. Using Compute Shaders instead of performing the computations on the CPU provides a significant speed up since they are able to be computed in parallel on the GPU.

There are, however, some limitations when working with Compute Shaders. For starters, the entirety of the algorithm cannot simply by copied and pasted into one Compute Shader and work correctly. While these computations are done in parallel only a few of them can be done in

lockstep, with that number being hardware dependent. Since the GPU allocates the computations the user has little control over the how the computations are completed or the order in which they are computed. So if a particular block of code needs to be completed by all inputs before moving onto something else, that block of code needs to be in its own shader. This is especially crucial for any global data that needs to be accessed in parallel between the threads or used for future computations.

C++ to GLSL Compute Shader

Transferring the 3D CLAHE algorithm from C++ to GLSL Compute Shaders needs to be done step by step. The first step in the algorithm is computing the LUT to map from HDR to SDR. The Equation for calculating this mapping is straightforward (Equation 5.1). However, in order to compute this table, we need to need to know what the max and min values of the volume are. In Python, this was calculated with a function call. In C++, a double nested for loop is needed to loop over every single pixel in the volume. With Compute Shaders, it is calculated with one dispatch call to run a single Compute Shader. The Compute Shader used to calculate the Max and Min values of a volume can be seen in Listing 5.1.

To use and run this shader a couple things need to happen first. As mentioned, the user needs to define the shader inputs and outputs. Data is inputted and outputted through Textures, Shader Storage Buffers, and Uniform variables. Textures are basically images or volumes, Shader Storage Buffers are used to store and receive data, and Uniform variables are constants that have been defined outside the shader. Defining these inputs and outputs inside the shader can be seen on lines 6, 9-11, and 13. This code is setting up where the GPU can access or store these different bits of information. In order for them to actually be there for the GPU to use, the C++ code needs to initialize and set up these objects. Once these objects are initialized they can then be passed to the GPU. Lines 3-5 in Listing 5.2 shows the calls needed to pass the inputs and outputs to the GPU and shader.

```
#version 430

layout(local_size_x = 4, local_size_y = 4, local_size_z = 4) in;

// input Dicom volume
layout(r16ui, binding = 0) uniform uimage3D volume;

// output calculation of the min/max values of the volume
layout(std430, binding = 1) buffer minMaxBuffer {
    uint[2] minMaxData;
};

uniform uint NUM_BINS;


void main() {

    // calculate the max and min for the volume
    uvec3 index = gl_GlobalInvocationID.xyz;
    uint val = imageLoad(volume, ivec3(index + offset)).x;
    uint maskVal = imageLoad(mask, ivec3(index + offset)).x;

    atomicMin(minMaxData[0], val);
    atomicMax(minMaxData[1], val);
}
```

Listing 5.1: GLSL Compute Shader to calculate the Max and Min Values of a Volume.

```
glUseProgram(_minMaxShader);
glBindImageTexture(0, _volTexture, 0, GL_TRUE, 1, GL_READ_ONLY, GL_R16UI);
glBindBufferBase(GL_SHADER_STORAGE_BUFFER, 1, globalMinMaxBuffer);
glUniform1ui(glGetUniformLocation(_minMaxShader, "NUM_BINS"), _numBins);

glDispatchCompute(  (GLuint)((volDims.x + 3) / 4),
                    (GLuint)((volDims.y + 3) / 4),
                    (GLuint)((volDims.z + 3) / 4));

glMemoryBarrier(GL_SHADER_IMAGE_ACCESS_BARRIER_BIT |
   GL_TEXTURE_FETCH_BARRIER_BIT);
glUseProgram(0);
```

Listing 5.2: Part of the C++ code needed to initialize and call the Compute Shader shown in Listing 5.1

These two pieces need to be in place in order for the shader to work as intended. To actually run a particular shader, it needs to be dispatched through the dispatch call starting on line 7 of listing 5.2. This is the call that actually starts executing the shader. The arguments inputted in this call along with the `local_size` variables in the corresponding Compute Shader indicate how many times to run the shader and how to break up those computations. These two code sections make sure that each pixel of the volume is processed. When a Compute Shader is dispatched, it will dispatch the number of groups indicated in the dispatch call. Each of these groups will be the size indicated in the shader, calculated as follows:

$$groupSize = \texttt{local\_size\_x} \cdot \texttt{local\_size\_y} \cdot \texttt{local\_size\_z}.$$

The GPU takes these local groups and attempts to execute each of them in parallel by distributing the work across a large array of GPU threads or lanes, generally referred to as a wave. While the number of threads, or the size of the waves, varies between different manufactures and models, common amounts are 64 and 32. It is generally more efficient to match the local size to the number of GPU threads available (wave size), just as it is more efficient to spawn the same number of threads as cores on the CPU. This is why I chose a work group size of 64. The last important aspect of the shader is that, since 64 threads are computing the Max and Min values in a group, it is important to make use of atomic operations to avoid race conditions. Memory barriers are set between shader calls to make sure the GPU schedules the computations in the desired order.

Once the Max and Min values for the volume are computed, a separate Compute Shader can then be called to compute the LUT. While the local size used by each shader should be similar to the wave size, the number of groups dispatched will vary depending on what is being calculated. To calculate the Max and Min pixel values, it was necessary to process the entire 3D volume. So the dispatch call created 3D groups based on the volume dimensions. For the LUT, a value needs to be calculated for each index, so the dispatch call and local size declared in the

shader will change. The LUT is a 1D array with length equal to the number of gray values in the original image. The dispatch call and local size declared in the shader to compute the LUT can be seen below.

```
glDispatchCompute( (GLuint)(numInputGrayValues / 64), 1, 1);
layout(local_size_x = 64, local_size_y = 1, local_size_z = 1) in;
```

Computing the Histograms follows a similar pattern to that of calculating the Max and Min values of the volume in terms of setting up and dispatching the shaders, and can be done with a single shader. Clipping the histograms and re-distributing the pixels is less straightforward. The algorithm works by evenly re-distributing the pixels that are above the clip value, which means that the amount of pixels above this value, or the number of excess pixels, needs to be known before they can be re-distributed. As a result, the process of clipping the histogram needs to be done through multiple shader executions - the first to calculate the number of excess pixels in each histogram, and a second to re-distribute those evenly throughout the histogram.

The next step is to calculate the mapping from the original gray value to the new Histogram Equalized gray value. This is done by calculating the Cumulative Distribution Function for each histogram. This is the only portion of the algorithm that I was unable to optimize through Compute Shaders. To be done efficiently, Cumulative Distribution Functions should be calculated in a sequential order. This is, unfortunately, not easy to make efficient with Compute Shaders. After dispatching the Compute Shader, there is no explicit control over the order in which those groups, and therefore indices, are being processed. So it is possible that the CDF value for index 100 gets computed before the value for index 1. This is a problem because the value for index 100 is dependent on the values calculated for index 1 through index 99. To ensure that these values were calculated sequentially, they were instead computed via CPU based multi-threading, with one thread per histogram.

The last and final step is interpolating between these mappings to generate the final 3D enhanced volume. Since this requires processing each pixel in the entire volume, this call

is similar to computing the Max and Min values and the initial histograms. Only instead of outputting data in Shader Storage Buffers, this shader outputs the enhanced 3D volume texture that can then be viewed. Overall, changing from Python to Compute Shaders decreased the computation time to the point that these could be completed in real-time. This will allow for the user to vary the parameters and interact the results in real-time.

Chapter 6

Results

6.1 Run-time Results

The initial implementation in Python was prohibitively slow for a real-time application, even in the 2D case, so it was in desperate need of optimization. The differences in language speed alone can clearly be seen through the speed up in run-times for the Python and C++ implementations. The improvement between C++ and the GLSL Compute Shader implementation, while not as drastic, does offer a significant boost in getting the algorithm to run in real-time. The average measured run-times for the Python, C++ and GLSL implementations can be seen in Table 6.1. These times are averaged over multiple runs with the exception of the Python 3D version. They were calculated on a laptop with an Intel HD Graphics 620 Graphics Card.

Table 6.1: The average computation time in seconds for the Python, C++ and GPU based CLAHE methods.

	Python	C++	GLSL
2D	2.201	0.025	–
2D focused	3.228	0.019	–
3D	400	3.092	1.165
3D focused	–	0.333	0.189

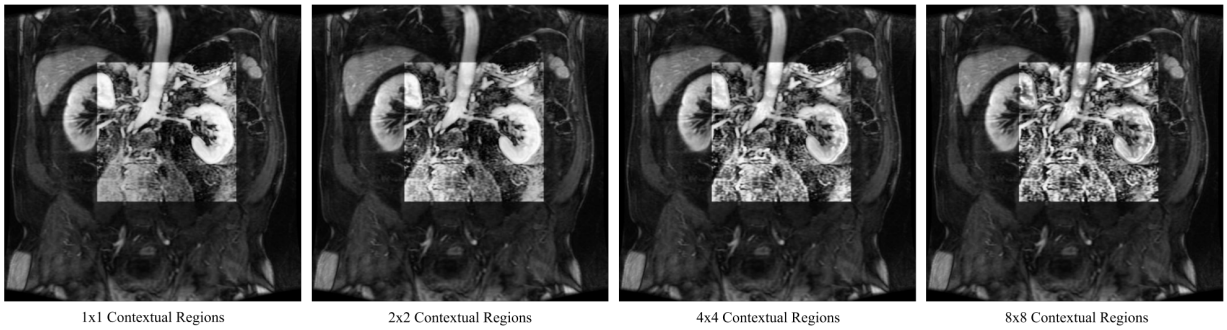


Figure 6.1: Comparing the number of Contextual Regions for a focused region in a DICOM slice.

6.2 Extensions

In an effort to provide more focused and varied control of the contrast enhancement, I implemented two new versions of a selective application of the 3D CLAHE algorithm; Focused CLAHE and Masked CLAHE.

6.2.1 Focused CLAHE

Focused CLAHE allows the user to focus the contrast enhancement on a particular region of the image or volume. This is accomplished by extracting the desired region from the original image or volume, and applying the CLAHE algorithm on just that extracted region. When working with a smaller portion of the entire image or volume, an important aspect to consider is the number of Contextual Regions or Sub-Blocks to use. The more regions used, the more specialized those mappings become which further enhances the images, sometimes to the point that any noise in the original image is also enhanced. The affect of just varying the number of Contextual Regions can be seen in Figure 6.1. All these results are applied to the same $240x240$ pixels with a clip limit of 0.75. With all other factors held constant, it can be seen that the increased number of Contextual Regions also enhances the noise.

To help reduce the amount of noise in the final image, instead of letting the user choose the number of Contextual Regions in addition to the area CLAHE is applied to, I decided to calculate the number of Contextual Regions or Sub-Blocks to use based on number of pixels in

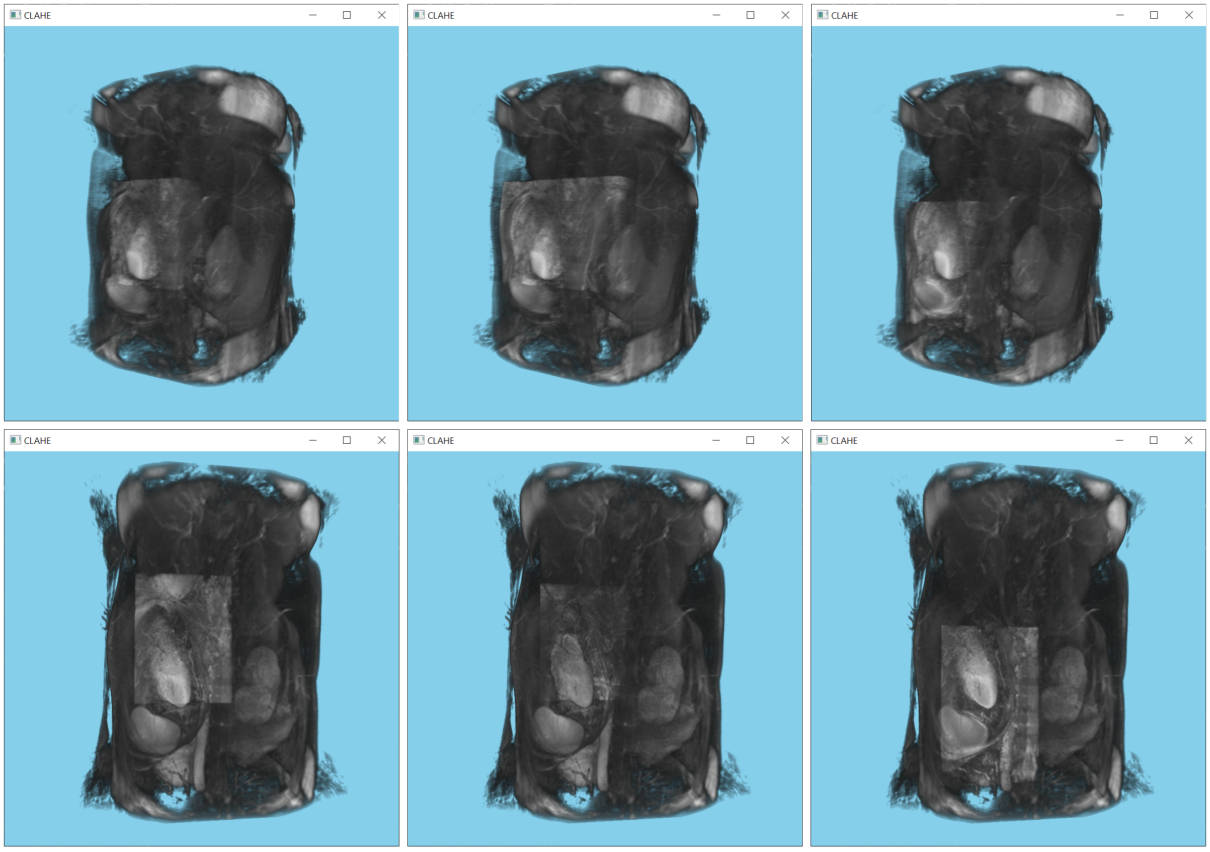
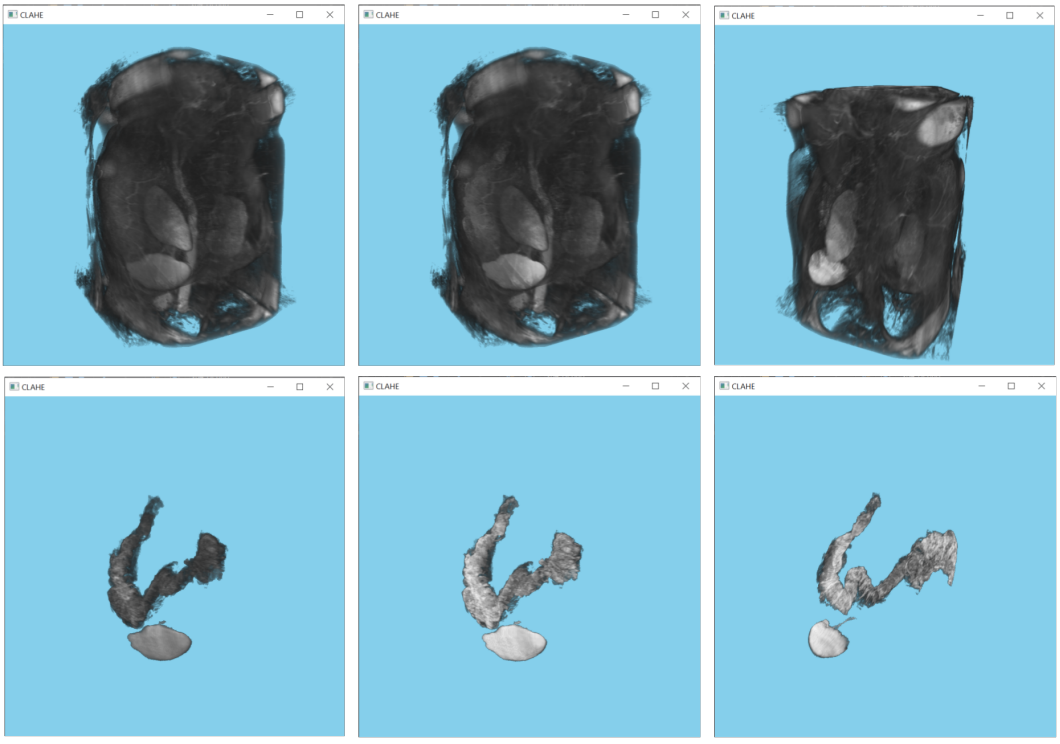


Figure 6.2: Different views of the 3D Focused CLAHE with varying focused regions.

the desired focused region. Based on visual inspection and trial and error, I found that using one Contextual Region or Sub-Block for every 100 pixels seems to produce a good balance between adaptive enhancement without too much noise. The results from the 3D version of this Focused CLAHE method can be seen in Figure 6.2. The size and placement of the focused region can be manipulated by the user, with the results shown in real-time. The dispatch calls to create the histograms for Focused CLAHE are based only on the size of the region. However, the call to interpolate and create the entire volume still needs to be based on the full volume dimensions to produce a complete volume.



Raw DICOM Masked CLAHE Masked CLAHE

Figure 6.3: Comparison between the raw DICOM volume and the Masked CLAHE version. Shown both as a part of the entire volume and by itself.

6.2.2 Masked CLAHE

A second method to focus the enhancement on a region of interest is Masked CLAHE. In this case, the enhancement is directed to a particular organ or set of organs. This method utilizes masks of the organs for each slice in the DICOM to help determine which pixels to include in the histograms and apply the enhancement to. In Focused CLAHE, the volume is divided into Sub-Blocks, with the number of Sub-Blocks based on the size of the focused region. In the Masked version, CLAHE is applied to a small region of the overall volume. This means that breaking that masked region into Sub-Blocks runs the risk of having a very small number of pixels in each histogram, or not having those Sub-Blocks be evenly distributed throughout the organs. As a result, I chose to implement this Masked version with just one Sub-Block for each masked organ.

Chapter 7

Conclusion

The goal of this thesis was to develop and implement an algorithm that would help improve the contrast of MRI data on an 8 bit display in a Virtual Reality application. In order to accomplish this, I used a 3D version of CLAHE. CLAHE itself is geared towards improving the contrast of medical imaging, and the 3D version further improves that enhancement for medical volumes. I optimized the 3D CLAHE computations for real-time interaction by computing them with GLSL Compute Shaders to take advantage of the parallel computations on the GPU. I then extended 3D CLAHE to develop different ways of enhancing the contrast in a particular region of the volume - Focused CLAHE, in which the user can choose a specific region of the volume to apply the contrast enhancement, and Masked CLAHE, in which the user can enhance the contrast of a particular organ. With the GPU based implementation, all of these computations can be done in real-time to enable the user to interact with the different methods. While this does mark an improvement in the ability to visualize medical volumes in 3D, there are still places for improvement and future work.

7.1 Future Work

Currently the algorithm uses a linear Look Up Table (LUT) to map the High Dynamic Range values to the Standard Dynamic Range. The simplicity of this method is beneficial to the overall speed of the algorithm. However, it is possible to improve on this by utilizing a better compression method such as the Accelerated Bilateral Filter method [7]. However, any enhancements in the HDR compression would need to be capable of maintaining the real-time interaction of the entire method.

Changes to the underlying CLAHE algorithm could be needed if the doctors determine that the results are lacking. If the doctors determine that the enhancement is increasing the contrast to the point that the results have lost needed detail, then the Local Contrast Modification CLAHE method could be added to help improve the results [29]. Additionally the current method used in Masked CLAHE is not adaptive. Further work should be done in determining if this is the best approach, or if the results could be improved with the addition of the Adaptive methods and using more than one Contextual Region.

Ultimately this algorithm was designed to help visualize the details in medical volumes, so feedback from the medical professionals who will be using these results is needed to gain a better understanding of where to focus the next steps and future work.

Bibliography

[1] O. Alshaya, Y. Pengpeng, R. Ni, Y. Zhao, and A. Piva. A new dataset for source identification of high dynamic range images. *Sensors*, 18:3801, November 2018.

[2] P. Amorim, T. Moraes, J. Silva, and H. Pedrini. 3d adaptive histogram equalization method for medical volumes. In *VISIGRAPP*, 2018.

[3] H. Cheng and X. Shi. A simple and effective histogram equalization approach to image enhancement. *Digital Signal Processing*, 14:158–170, March 2004.

[4] P. E. Debevec and J. Malik. Recovering high dynamic range radiance maps from photographs. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques*, SIGGRAPH '97, pages 369–378, New York, NY, USA, 1997. ACM Press/Addison-Wesley Publishing Co.

[5] J. DiCarlo and B. Wandell. Rendering high dynamic range images. *Proceedings of SPIE - The International Society for Optical Engineering*, 3965:392–401, May 2000.

[6] R. Dorothy, J. R M, J. Rathish, S. Prabha, S. Rajendran, and S. Joseph. Image enhancement by histogram equalization. *International Journal of Nano Corrosion Science and Engineering*, 2:21–30, September 2015.

[7] F. Durand and J. Dorsey. Fast bilateral filtering for the display of high-dynamic-range images. *ACM Trans. Graph.*, 21(3):257–266, July 2002.

[8] M. Eramian and D. Mould. Histogram equalization using neighborhood metrics. In *The 2nd Canadian Conference on Computer and Robot Vision (CRV'05)*, pages 397–404, May 2005.

[9] E. L. Hall. Almost uniform distributions for computer image enhancement. *IEEE Transactions on Computers*, C-23:207–208, 1974.

[10] J.-H. Han, S. Yang, and B.-U. Lee. A novel 3-d color histogram equalization method with uniform 1-d gray scale histogram. *Trans. Img. Proc.*, 20(2):506–512, February 2011.

[11] J. E. J. M. Hugo Mayo, Hashan Punchihewa. Types of medical imaging.

[12] Y.-T. Kim. Contrast enhancement using brightness preserving bi-histogram equalization. *IEEE Trans. on Consum. Electron.*, 43(1):1–8, February 1997.

[13] T. T. Kimpe, Tom. Increasing the number of gray shades in medical display systems—how much is enough? *Journal of Digital Imaging*, 20(4):422–432, December 2007.

[14] G. W. Larson, H. Rushmeier, and C. Piatko. A visibility matching tone reproduction operator for high dynamic range scenes. *IEEE Transactions on Visualization and Computer Graphics*, 3(4):291–306, 1997.

[15] K. W. Leszczynski and S. Shalev. A robust algorithm for contrast enhancement by local histogram modification. *Image and Vision Computing*, 7(3):205–209, August 1989.

[16] M. Li, D. Wilson, M. Wong, and A. Xthona. The evolution of display technologies in pacs applications. *Computerized medical imaging and graphics : the official journal of the Computerized Medical Imaging Society*, 27:175–84, February 2003.

[17] L. Lu, Y. Zhou, K. Panetta, and S. Agaian. Comparative study of histogram equalization algorithms for image enhancement. In S. S. Agaian and S. A. Jassim, editors, *Mobile Multimedia/Image Processing, Security, and Applications 2010*, volume 7708, pages 337 – 347. International Society for Optics and Photonics, SPIE, 2010.

[18] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. In *Proceedings. 1999 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (Cat. No PR00149)*, volume 1, pages 374–380 Vol. 1, 1999.

[19] S. Muniasamy, K. Ramar, N. Arumugam, and G. Prabin. Histogram modified local contrast enhancement for mammogram images. *Int. J. of Biomedical Engineering and Technology*, 9:60 – 71, January 2012.

[20] W. Mustafa and M. Kader. A review of histogram equalization techniques in image enhancement application. *Journal of Physics: Conference Series*, 1019:012026, June 2018.

[21] S. K. Nayar and T. Mitsunaga. High dynamic range imaging: spatially varying pixel exposures. In *Proceedings IEEE Conference on Computer Vision and Pattern Recognition. CVPR 2000 (Cat. No.PR00662)*, volume 1, pages 472–479 vol.1, 2000.

[22] C. H. Ooi and N. A. Mat Isa. Quadrants dynamic histogram equalization for contrast enhancement. *IEEE Transactions on Consumer Electronics*, 56(4):2552–2559, 2010.

[23] R. B. Paranjape, W. M. Morrow, and R. M. Rangayyan. Adaptive-neighborhood histogram equalization for image enhancement. *CVGIP: Graph. Models Image Process.*, 54(3):259–267, May 1992.

[24] S. H. Pizer, J. D. Austin, J. R. Perry, H. D. Safrit, and J. J. Zimmerman. Adaptive histogram equalization for automatic contrast enhancement of medical images. In *Other Conferences*, 1986.

[25] S. M. Pizer, E. P. Amburn, J. D. Austin, R. Cromartie, A. Geselowitz, T. Greer, B. T. H. Romeny, and J. B. Zimmerman. Adaptive histogram equalization and its variations. *Comput. Vision Graph. Image Process.*, 39(3):355–368, September 1987.

[26] S. M. Pizer, R. E. Johnston, J. P. Ericksen, B. C. Yankaskas, and K. E. Muller. Contrast-limited adaptive histogram equalization: speed and effectiveness. In *Proceedings of the First Conference on Visualization in Biomedical Computing*, pages 337–345, May 1990.

[27] Qian Wang, Liya Chen, and Dinggang Shen. Fast histogram equalization for medical image enhancement. In *2008 30th Annual International Conference of the IEEE Engineering in Medicine and Biology Society*, pages 2217–2220, 2008.

[28] S. Raj, S. Raj, and S. Kumar. An improved histogram equalization technique for image contrast enhancement. In *ICC-2015*, July 2015.

[29] S. Sajeev and M. Ravishankar. Modified contrast limited adaptive histogram equalization based on local contrast enhancement for mammogram images. In *Communications in Computer and Information Science*, volume 296, April 2012.

[30] J. Stark. Adaptive image contrast enhancement using generalizations of histogram equalization. *IEEE Transactions on Image Processing*, 9(5):889–896, May 2000.

[31] V. Stimper, S. Bauer, R. Ernstorfer, B. Schölkopf, and R. P. Xian. Multidimensional contrast limited adaptive histogram equalization, 2019.

[32] P. E. Trahanias and A. N. Venetsanopoulos. Color image enhancement through 3-d histogram equalization. *Proceedings., 11th IAPR International Conference on Pattern Recognition. Vol. III. Conference C: Image, Speech and Signal Analysis,*, pages 545–548, 1992.

[33] M. Zahedi and M. Rahimi. 3d color histogram equalization by principal component analysis. *J. Vis. Comun. Image Represent.*, 39(C):58–64, August 2016.

[34] H. Zhu, F. H. Chan, and F. Lam. Image contrast enhancement by constrained local histogram equalization. *Comput. Vis. Image Underst.*, 73(2):281–290, February 1999.

[35] K. Zuiderveld. *Contrast Limited Adaptive Histogram Equalization*, page 474–485. Academic Press Professional, Inc., USA, 1994.