Real-Time Contrast Enhancement for 3D Medical Images using Histogram Equalization

Karen Lucknavalai and Jürgen P. Schulze

University of California San Diego

Abstract. 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 paper introduces a real-time implementation of 3D Contrast Limited Adaptive Histogram Equalization (CLAHE) 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 on 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.

Keywords: Volume rendering · Medical imaging · High dynamic range.

1 Introduction

Medical imaging, such as computed tomography (CT) and magnetic resonance (MR) imaging is critical when it comes to helping doctors diagnose patients and prepare for surgeries. Diagnoses are typically done by radiologists, while surgeries are planned by surgeons. Radiologists typically view and study medical images one slice at a time on special high-contrast 2D monitors. Surgeons use the diagnosis they are given by the radiologist, and sometimes also look at the medical image stacks, on their own computers which are standard laptop or desktop computers with regular 2D monitors. Given that viewing 2D image stacks from the CT and MR scans on a monitor 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, and at the highest image quality their displays can reproduce.

The reason radiologists use special high contrast monitors is that CT and MR data are typically generated with a single luminance data point per pixel at 12 bit dynamic range. Standard monitors only have a dynamic range of 8 bit for luminance values, which are typically displayed as grayscale values.

The goal for our work was to maximize the available contrast in the data for virtual reality (VR) headsets. These often have an even smaller dynamic range than desktop monitors, which made the work even more important for us. But the findings in this paper apply to regular monitors equally well, as long as the goal is to display the CT or MR image stack in its entirety with volume rendering techniques.

This paper offers three novel contributions. First it introduces a real-time implementation of 3D Contrast Limited Adaptive Histogram Equalization (CLAHE). This allows medical professionals to interactively enhance the contrast of a medial volume or image stack within a 3D viewer. 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.

2 Related Work

2.1 Medical Images

MR scanner data is stored in DICOM files as a series of 12 bit grayscale images with each slice of the scan stored as a separate image. In order to display as much of the detail as possible, special high dynamic range (HDR) grayscale monitors were developed. These monitors are based on the Grayscale Standard Display Function, which was developed based on the number of gray values the human eye can detect [5, 7].

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 allow a user to view an MRI scan 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 is the same as their 2D counterparts. There have been some more recent developments using virtual and augmented reality (AR) to display and interact with 3D scans in an immersive environment. However, they are still in development and are all limited to displaying within the 8 bit standard dynamic range. While HDR grayscale displays found a hardware solution to display the details in medical scans, the same solution is not currently available on laptops or in AR and VR headsets. Overcoming this mismatch, and enhancing the contrast of the 3D visualization of these medical images in real-time is the focus of this paper.

2.2 Contrast Enhancement Techniques

Image processing techniques have been around for decades. One method that has sparked a lot of research is histogram equalization (HE) [4]. This method was developed in the 1970s and has the goal of increasing the global contrast of an image, and is especially useful when applied to images that are either over or under exposed. HE works by redistributing pixel values for an image throughout the entire dynamic range, and has been shown to produce particularly good results in x-ray images [2].

Histogram equalization enhances the contrast based on the pixel distribution of the entire image, which may lead to areas that still appear over or underexposed. Adaptive Histogram Equalization (AHE) was developed to help combat this effect and improve the contrast of these local regions. AHE creates local histograms for each of the pixels in an image, so that the final enhancement adapts to these smaller image sections. This is done by dividing the image into Contextual Regions (CR) and bi-linearly interpolate between these local histograms to generate the final image [6, 9].

A drawback to both Histogram Equalization and Adaptive Histogram Equalization is the possibility of magnifying noise within the original image. Contrast Limited Histogram Equalization aims to counteract this problem by limiting the contrast amplification. This is done by clipping the histogram at a predefined value, or clip limit, which in turn reduces the final contrast of the image [9]. Contrast Limited Adaptive Histogram Equalization (CLAHE) combines the advantages of the Contrast Limited approach to limit contrast and noise, as well as the ability to decrease the over and underexposed regions in the final image with Adaptive Histogram Equalization [12].This combination of methods provides multiple parameters that allow for flexibility and control over the final enhancement of the image. Adjusting the number of CRs 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.

2.3 Medical Imaging Enhancement

Since Histogram Equalization is both an effective and simple method for contrast enhancement, it is a popular choice to enhance medical images. CLAHE has a track record for providing good contrast enhancements for a variety of medical images, and has been shown to be effective and helpful for the doctors diagnosis, and interaction with the medical images [10, 8]. With the development and availability of 3D visualization, multiple papers have taken to adapting 2D methods to 3D. Amorim [1] extended CLAHE for 3D volumes by creating local histograms for a sub-volume of the medical volume, and tri-linearly interpolating between these local histograms. These results while impressive are not fast enough for real-time applications. This work takes this 3D CLAHE method and develops it for real-time applications.

3 Implementation

We started by developing a Python version of the original 2D CLAHE algorithm, and the 3D extension by Amorim [1, 12]. The first step in 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 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. This LUT will be the size of the input image dynamic range and contain the mappings between the original dynamic range to the desired dynamic range. This LUT and the mapping used to generate it is where the dynamic range disparity is handled.

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

After this LUT is generated, the local histograms are created. This is done by dividing the image or volume into smaller sections or CRs. Local histograms are then created for these sections where each histogram counts the number of times each of the possible gray values occur within a particular CR. 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 CR are redistributed evenly throughout that histogram. This redistribution limits the amount of times a particular gray value can occur, 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 [12] was calculated as given in Equation 2. This one $clipValue$ 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 CRs.

$$clipValue = clipLimit \cdot \frac{(sizeCRx \cdot sizeCRy)}{numBins} \quad , \; cliLimit \in [1, \infty) \tag{2}$$

The $clipLimit$ is a user inputted value that must be larger than 1. A value of 1 corresponds to a completely uniform distribution of pixels and visually results in an unaltered image. As the clip limit increases, the contrast is also increased. If, however, the $clipValue$ is larger than the count for the most common gray value in a CR, the histogram will not be clipped and the clip Limit will have no effect on the image. Figure 1 shows the histograms and resulting images for the example case of using just one CR for the image. 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 CRs 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. We implemented

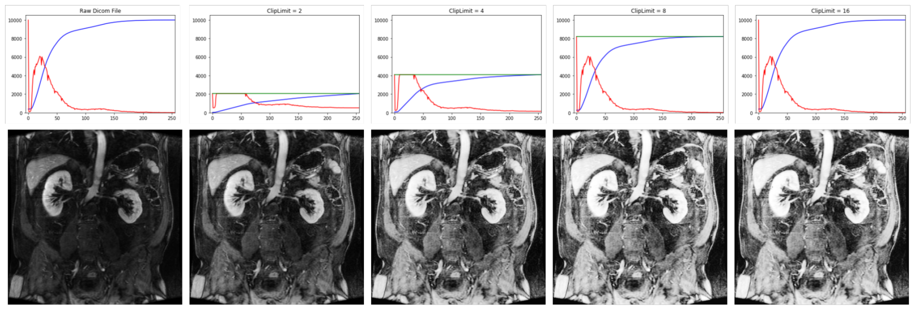

Fig. 1. Resulting Images and plots of the histogram (in red), CDF (in blue), and $clipValue$ (in green). Using a single Contextual Region for the entire image and varying values for the $clipValue$ calculated with Equation 2.

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 CR (Equation 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 $clipValue$ since it would be impossible to re-distribute the values of the histogram if they were clipped below a uniform distribution. The $minClipVal$ or lower bound is found utilizing the original formula for the $clipValue$ and 1.1 as the clip limit because the closer the $clipValue$ is to a uniform distribution, the harder it gets to re-distribute the histogram values. Figure 2 shows the results of this approach for varying clip values on a DICOM slice, again using 4x4 CRs.

$$\begin{aligned} clipValue &= max\big(minClipVal,\ \ clipLimit \cdot max(hist_{currCR})\big) \\ minClipVal &= 1.1 \cdot \frac{(sizeCRx \cdot sizeCRy)}{numBins},\ \ clipLimit \in [0,1] \end{aligned} \tag{3}$$

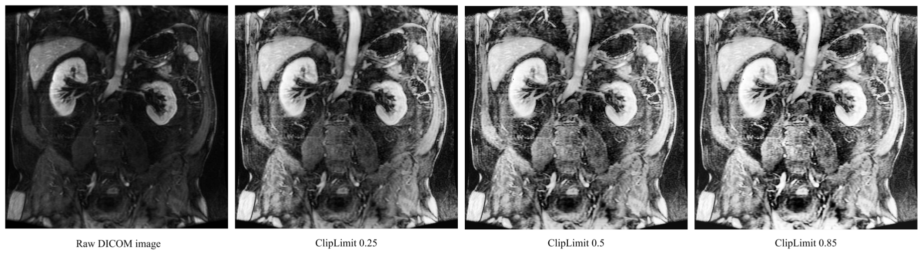

Fig. 2. Applying CLAHE to a slice of an MRI DICOM with an increasing $clipLimit$ where $clipValue$ is calculated with Equation 3.

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 3 shows the comparison between these two approaches using 4x4 CRs 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 will be using this local approach to the clip limit.

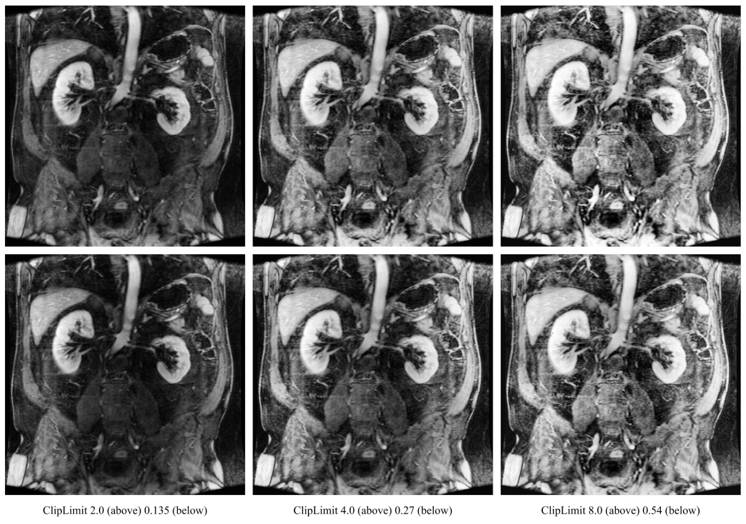
ClipLimit 2.0 (above) 0.135 (below)
ClipLimit 4.0 (above) 0.27 (below)
ClipLimit 8.0 (above) 0.54 (below)

Fig. 3. Comparison between using the $clipValue$ as calculated in the original CLAHE paper (Equation 2, with the presented adaptive/local approach (Equation 3).

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. The final pixel value is determined by bi-linearly interpolating between the 4 neighboring CR mappings in the 2D case, and tri-linearly interpolating between the 8 neighboring CR mappings in the 3D case. The results of the 3D implementation of CLAHE can be seen in Figure 4. All the 2D images shown are from DICOM slices which are 512x512 pixels, and the Volumes are on the full 512x512x116 DICOM volume.

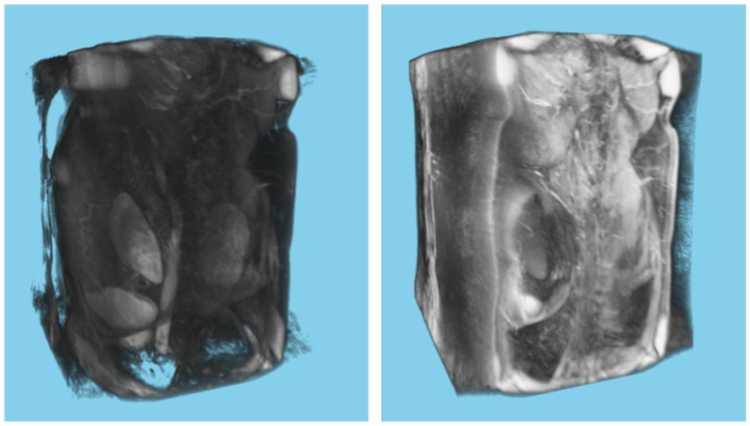

Fig. 4. Comparison between the original DICOM volume on the left, alongside the CLAHE enhanced volume on the right.

3.1 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 change alone produced a considerable improvement: computing the entire volume went from 5 minutes to 3 seconds, which was still not fast enough for real-time. To further optimize, we transferred the algorithm onto the GPU using GLSL Compute Shaders.

Using GLSL Compute Shaders provides a significant speed up since they are computed in parallel on the GPU. But only a few of them can be done in lockstep due to hardware constraints. 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, it is safest to place that block of code 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.

The first step in the 3D CLAHE algorithm is computing the LUT to map from HDR to SDR. The equation for calculating this mapping is straightforward (Equation 1). However, in order to compute this table, we 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 that processes each pixel in the entire volume.

Once the Max and Min values for the volume are computed, a separate compute shader can then be called to compute the LUT. Computing the histograms can similarly 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 not optimized 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 process each pixel in the volume and generate the final 3D enhanced volume. 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.

4 Results

4.1 Run-time Results

The initial CLAHE implementation in Python was prohibitively slow for a real-time application, even in the 2D case, so it was in 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 1, and the data used was a 512x512x116 DICOM volume.

4.2 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.

Table 1. The average computation time in seconds for the Python, C++ and GPU based CLAHE methods, calculated on a laptop with an Intel HD Graphics 620 Graphics Card.

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

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, the number of CRs has a large impact on the results. 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 CRs can be seen in Figure 5. 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 CRs also enhances the noise.

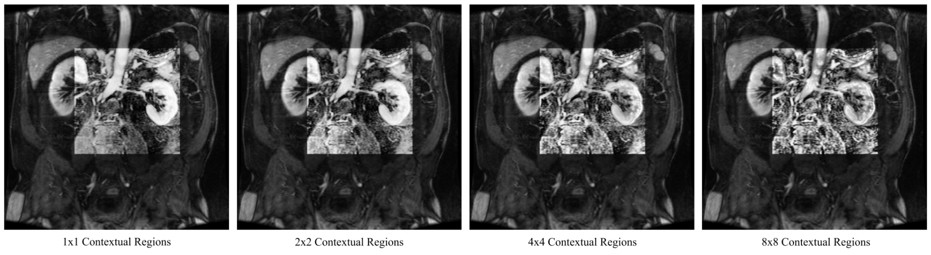

Fig. 5. Comparing the number of Contextual Regions for a focused region in a DICOM slice.

To help reduce the amount of noise in the final image, instead of letting the user choose the number of CRs in addition to the area CLAHE is applied to, we decided to calculate the number of CRs to use based on number of pixels in the desired focused region. Based on visual inspection and trial and error, we found that using one CR 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. 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.

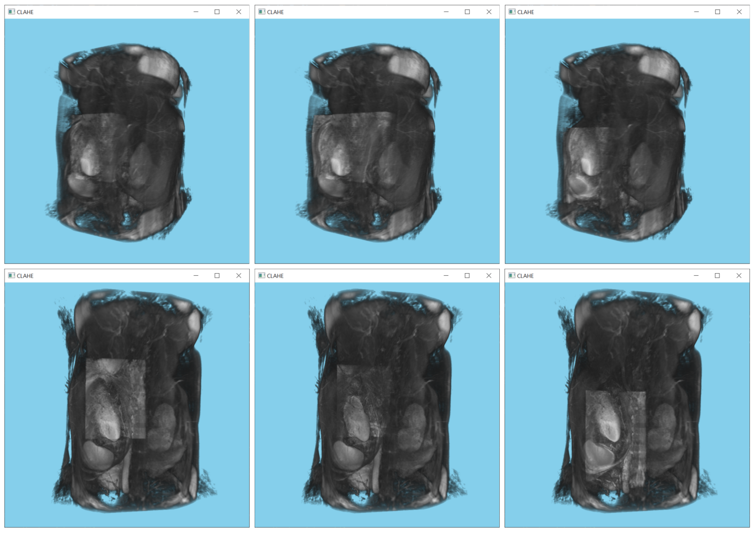

Fig. 6. Two different viewpoints of the DICOM volume with Focused CLAHE applied to different regions of the volume. These images show the flexibility of Focused CLAHE in that the user can adjust both the size and placement of the focused region.

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 CRs, with the number of CRs 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 CRs runs the risk of having a very small number of pixels in each histogram, or not having those CRs be evenly distributed throughout the organs. As a result, we chose to implement this Masked version with just one CR for each masked organ.

5 Conclusions

We developed an algorithm to help improve the contrast of MRI data on typical monitors and VR displays. To accomplish this, we developed 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

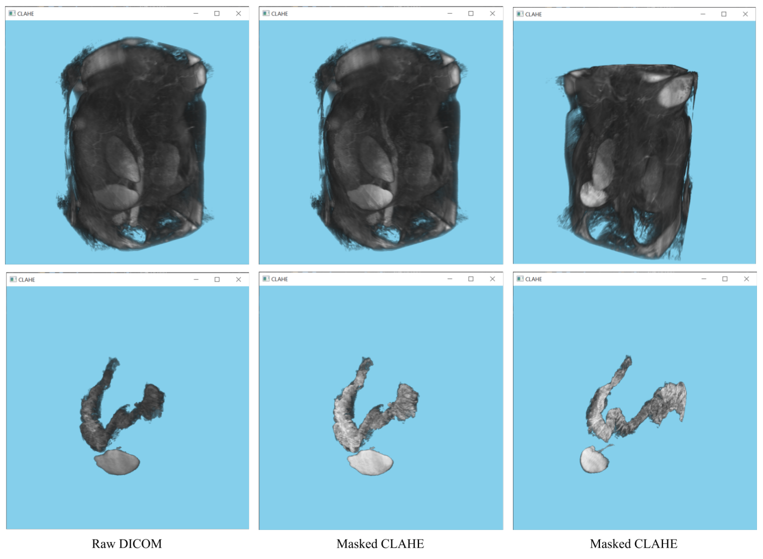

Fig. 7. The first column of images show the raw DICOM file displayed as a part of the entire volume, and the organs alone. The next four images show two different views of the DICOM volume and organs after applying Masked CLAHE.

volumes. We optimized the 3D CLAHE computations for real-time interaction by computing them with compute shaders. We then extended 3D CLAHE to focus on a particular region of the volume - Focused CLAHE, and Masked CLAHE, in which the user can enhance the contrast of a particular organ. With our GPU-based implementation, all of these computations can be done in real-time to enable the user to interact with the different methods.

6 Future Work

Our current algorithm uses a linear look-up table to map the HDR values to the standard dynamic range. The simplicity of this method is beneficial to the overall speed of the algorithm, but it may be possible to use a better compression method such as the Accelerated Bilateral Filter method [3].

To further improve the CLAHE algorithm, the Local Contrast Modification CLAHE method could be added to help improve the results [11]. The current method used in Masked CLAHE is not adaptive. It should be investigated if the results could be improved with the addition of adaptive methods and using more than one CR.

7 Acknowledgements

This research was funded in part by a grant from The Leona M. and Harry B. Helmsley Charitable Trust. We thank Trevor Hedstrom for his help with the implementation of the shader version of the algorithm.

References

1. Amorim, P., Moraes, T., Silva, J., Pedrini, H.: 3d adaptive histogram equalization method for medical volumes. In: VISIGRAPP (2018)
2. Dorothy, R., R M, J., Rathish, J., Prabha, S., Rajendran, S., Joseph, S.: Image enhancement by histogram equalization. International Journal of Nano Corrosion Science and Engineering **2**, 21–30 (September 2015)
3. Durand, F., Dorsey, J.: Fast bilateral filtering for the display of high-dynamic-range images. ACM Trans. Graph. **21**(3), 257–266 (July 2002). https://doi.org/10.1145/566654.566574
4. Hall, E.L.: Almost uniform distributions for computer image enhancement. IEEE Transactions on Computers **C-23**, 207–208 (1974)
5. Kimpe, T., Tuytschaever, T.: Increasing the number of gray shades in medical display systems—how much is enough? Journal of Digital Imaging **20**(4), 422–432 (December 2007). https://doi.org/10.1007/s10278-006-1052-3
6. Leszczynski, K.W., Shalev, S.: A robust algorithm for contrast enhancement by local histogram modification. Image and Vision Computing **7**(3), 205–209 (August 1989). https://doi.org/10.1016/0262-8856(89)90045-0
7. Li, M., Wilson, D., Wong, M., Xthona, A.: 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). https://doi.org/10.1016/S0895-6111(02)00072-1
8. Pizer, S.M., Johnston, R.E., Ericksen, J.P., Yankaskas, B.C., Muller, K.E.: Contrast-limited adaptive histogram equalization: speed and effectiveness. In: Proceedings of the First Conference on Visualization in Biomedical Computing. pp. 337–345 (May 1990). https://doi.org/10.1109/VBC.1990.109340
9. Pizer, S.M., Amburn, E.P., Austin, J.D., Cromartie, R., Geselowitz, A., Greer, T., Romeny, B.T.H., Zimmerman, J.B.: Adaptive histogram equalization and its variations. Comput. Vision Graph. Image Process. **39**(3), 355–368 (September 1987). https://doi.org/10.1016/S0734-189X(87)80186-X
10. Pizer, S.M., Austin, J.D., Perry., J.R., Safrit, H.D., Zimmerman, J.B.: Adaptive Histogram Equalization For Automatic Contrast Enhancement Of Medical Images. In: III, S.J.D., Schneider, R.H. (eds.) Application of Optical Instrumentation in Medicine XIV and Picture Archiving and Communication Systems. vol. 0626, pp. 242–250. International Society for Optics and Photonics, SPIE (1986). https://doi.org/10.1117/12.975399
11. Sajeev, S., Ravishankar, M.: Modified contrast limited adaptive histogram equalization based on local contrast enhancement for mammogram images. In: Communications in Computer and Information Science. vol. 296 (April 2012). https://doi.org/10.1007/978-3-642-35864-7_60
12. Zuiderveld, K.: Contrast Limited Adaptive Histogram Equalization, p. 474–485. Academic Press Professional, Inc., USA (1994)