5-2020

Image Enhancement and Restoration for Colonoscopy Images

Sarah Paracha
University of Arkansas, Fayetteville

Image Enhancement and Restoration for Colonoscopy Images

An Undergraduate Honors College Thesis

in the

College of Engineering
University of Arkansas
Fayetteville, AR
May, 2020

by

Sarah Paracha

Table of Contents

Abstract

Colonoscopy images contain specular highlights that occur as a result of the tiny camera on the colonoscope being perpendicular to the image location. These specular highlights may prevent the Gastroenterologist from having a full picture of the patient's condition and potentially giving an early diagnosis. The purpose of my honors research is to remove the specular highlights from these colonoscopy images.

The process to achieve the above objective involves two steps. The first step would require locating the specular highlights in the image through image segmentation. For this purpose, information from nearby x and y pixels may be utilized. The second step consists of using image restoration to fill in the specular regions. The removal of these specular highlights refines the colonoscopy image and allows useful information to be deduced by the physician.

1. Introduction

Medical images are used by physicians to examine the patients' internal anatomy and gain more information about potential complications and/or patient symptoms. For instance, magnetic resonance images (MRI) and computed tomography (CT) images can provide knowledge on the location of tumor growths. This knowledge appears useful to the surgeon in ensuring that no cancerous cells are left behind during removal of the tumor. However, medical images are likely to contain inaccuracies in the form of noise. Therefore, image processing can play a pivotal role in creating refined images that may ensure vital information is not disclosed under noise and a physician can take the best course for diagnosis of the medical condition [1].

Medical image processing consists of two primary divisions i.e. segmentation and restoration. Segmentation apportions the image into areas that are of relevance [2]. Image segmentation can further be categorized as region or non-region based [3]. However, the uniqueness of various medical images makes it difficult to designate a single image segmentation method as most applicable. Instead each medical image type and its purpose needs to be considered when choosing an algorithm for segmentation [4].

For the purpose of analyzing colonoscopy images, two image segmentation methods will be studied i.e. region growing and background subtraction. Region growing is a region-based segmentation method which takes into account information about local pixels [3]. Region growing leads to accumulation of pixels on the image into sections based on a designated condition. This accumulation begins on the selection of a seed point and examination of neighboring pixels [4]. On the other hand, background subtraction differentiates the image into two components i.e. the background and the foreground. By removing the background of the image, the focus is brought to identify objects on the foreground [5].

The second category of image processing is image restoration. Image restoration includes replacing the tarnished areas in medical images with hidden useful information or minimizing the noise occurrence in an image. Medical images are corrupted due to many reasons. An example being defects in the optical instruments. In the case of colonoscopy images, specular highlights are formed due to camera misalignment relative to the colon surface. It is a hard task to not only identify areas in the image that are corrupted but to also replace these areas without compromising the information provided by the image and its image quality [6]. This paper will briefly examine a simple approach for restoration of colonoscopy images and the effectiveness of the resulting images obtained.

2. Related Work

Variations in region growing have been proposed to segment realistic color images. One such variation includes using a single seed to region grow. This method sets the center pixel as the initial seed and uses conditional criteria to assign a pixel to the region or boundary of an image. These conditional criteria take into account both local and global properties of pixels in the image. It then proceeds to grow alternate regions in a hierarchical manner by selecting seed points from a stack data structure containing the boundary pixels. Growing the regions in this manner declines computational expense. Additionally, this method simplifies the initial seed point selection [3]. The use of region growing can also be expanded to medical images such as MRI scans for brain tumor segmentation. A promising approach for this includes using an adaptive model that may grow a region based on the variances and gradients along and internal to the boundary curve. Doing so aims to take advantage of the heterogeneity present amongst pixels in the tumor tissue and the edge outlining the tumor [4].

Background subtraction is commonly used in motion detection. For this purpose, background subtraction may be performed on a sequence of image frames where foreground would be the moving object such as a walking human. However, the nature of outdoor images used in motion detection can present an array of challenges. These challenges include the changing brightness of images and ensuring that irrelevant objects (e.g. flowers in the background) are not identified [5]. Moreover, the application of background subtraction can be extended to medical images produced such as images generated from Medical Infrared Thermography (MIT). For instance, utilizing background subtraction to segment a knee thermogram can be helpful in detecting knee abnormality and/or arthritis. Doing so indicated increased accuracy in disease identification and led to a decline in under segmentation [7].

In the area of medical image restoration, there are many methods that may be utilized. Current review indicates that sparse representations are thought to be promising in comparison to other representations [6].

From current literature, it may be deduced that medical image processing is a dynamic area of research. Research in this area plays a significant role in healthcare as it could equip physicians with essential information for patient care. In addition, medical image processing techniques are not uniform for all medical images. Therefore, this paper aims to add knowledge to the existing community with regards to image processing of colonoscopy images.

3. Using Region Growing to Identify Specular Highlights

3.1 Algorithm Design

As an image segmentation method, region growing segregates the image into clusters that share similarities. The use of region growing to identify specular highlights in colonoscopy images relies on the principle that the pixels located in these regions will share the same brightness. Therefore, region growing would allow pixels with the same brightness value to be grouped together.

To grow the regions an initial pixel point needs to be chosen. This pixel point is known as the seed point. The seed point itself must be a pixel with brightness values close to that of pixels in the specular highlight region so that only the specular target regions can be grown. Thus, the initial approach taken for seed point selection depended on the use of local extrema values primarily the local maximum. This is because the local maximum values in the color image will have brightness values close to that of those in specular highlight regions (i.e. values closer to the RGB white color value of 255).

However not every local maximum could be chosen as a seed point. This is because present in colonoscopy images are shades of lighter pink from the colon surface. These pixels may be considered a local maximum comparative to other pixels located in the same area on the image, but they are not bright enough to be considered a part of a specular highlight. Thus, in order to determine whether a particular pixel could substitute as a seed point, *thresholding* was used. The objective of thresholding is to categorize the intensity range of an image into two classifications i.e. the background and the object using a threshold value. In this case the background would be the colon surface whereas the object would be the specular highlights. Therefore, all values falling below or equivalent to the threshold are rejected whereas all values greater than the threshold may be chosen as seed points.

The above approach for seed point selection was later modified to use *adaptive thresholding*. Adaptive thresholding introduces flexibility into the process as a threshold value is determined for each region of interest respectively. By doing so, each area of the image is examined independently for specular highlights thus ensuring that an overall rigid threshold value is not utilized. The idea behind this relies on the fact that a specular highlight in an image is identifiable due to not only its brightness value but also how bright it is amongst its surrounding pixels. This means that instead of examining the brightness values of pixels near the RGB color value for white, i.e. 255, pixel brightness was also assessed in comparison to neighbor pixel values.

After selecting a seed point, neighboring pixels surrounding the seed point pixels are examined. That is once a seed point is selected, the region is grown in the x and y direction. Upon examination of these neighboring pixels, a decision is made to either include the pixel in the region or to disregard it based on its similarity to the pixels in the growing region. All

specular highlights will not have the exact brightness value and may fall close to each other within a range. To allow for this flexibility, an acceptable variance was allowed from the threshold value for region growing. This variance would ensure that all possible specular region pixels are selected.

For the purpose of this project, we are not concerned with identifying the large specular highlight regions. Restoring the larger specular highlight regions proves to be more complex. This is because information from neighboring pixels may not necessarily be a reliable source to use in restoring the pixels in the larger specular region. For instance, in a worst case scenario, a larger specular highlight region may be hiding a tumor on the colon surface. Restoring it using neighboring pixel values from the colon surface may provide misinformation to the physician and can be harmful to the patient. Therefore, the larger specular highlight regions were flagged to be removed at the end of the program.

3.2 Implementation

The implementation of the region growing program involved the use of image processing libraries created by Dr. John Gauch, a professor in the Computer Science and Computer Engineering program at the University of Arkansas. These libraries contain four C++ classes (i.e. im_short, im_float, im_complex and im_color) used for defining image pixels and a plethora of functions useful for image processing. These functions were used in the implementation of this program.

The implementation of region growing to identify specular highlights can be described in several steps. The first step consists of reading the colonoscopy image in JPG format. After reading the image input into the program, a Gaussian filter was applied. The Gaussian filter blurs

the image and is effective in removing noise. Moreover, the Gaussian function places weight on the center pixels (i.e. provides a weighted average) and allows a deviation from the center of the bell curve equivalent to a chosen sigma value. This sigma value was obtained as an input to the program. Next a function is applied on the Gaussian image to determine the extrema values. An integer value is sent as an argument to the extrema function. This is the size of the neighborhood for which the local maximum values will be determined. This value is obtained as an input parameter to the program as well. The computed local maximum value will then replace every pixel in within the neighborhood size. Subsequently every pixel value is examined on the produced extrema image. For this, the image is scanned from left to right with the use of nested for loops. Scanning from left to right also ensures that the same seed point has not been chosen beforehand. Every pixel value is then compared to the input threshold value. This comparison to the threshold value then governs whether the pixel will be a seed point and if the region will be grown. An additional check is also performed by examining the created output image to ensure that the pixel point is not already a part of a grown region.

```
if ((Extrema.Data2D[y][x] > Threshold) && (Output.Data2D[y][x] == 0))
```

Figure 1: Seed Point Selection Using Thresholding

Subsequently, if a seed point is selected a call is made to the recursive function, *RegionGrowStack()*, on the Gaussian image. This function takes as input primarily the output image, the x and y coordinate positions, image pixel, region number, along with region size of the seed point. During the process of region growing a slight variation is allowed from the threshold. The amount of allowed variation is determined through an input parameter of the program sent as an additional argument to the *RegionGrowStack()* function.

```
Gaussian.RegionGrowStack(Output, x, y, total, size, MinDiff, color);
```

Figure 2: Region Growing Function Call

To implement adaptive thresholding, the above process included an additional step. To compare the brightness of pixel values to its neighbor pixels, the median brightness image was generated. This was performed through a call to the median function in the libraries with two integer arguments. This integer argument was predefined and would dictate the size of the median neighborhood. Seed point selection was then performed while scanning the image in iterative loops similar to the thresholding approach. However, this time a seed point was chosen by surveying how much brighter the image pixel value is in contrast to the respective median value.

```
if ((Extrema.Data2D[y][x] >  Median.Data2D[y][x] + Threshold) && (Output.Data2D[y][x] == 0))
```

Figure 3: Seed Point Selection Using Adaptive Thresholding

A minimum size and maximum size input was also obtained from the program. These values served as a window to govern the size of the specular highlights identified. The larger specular highlights were kept track of by storing them in an array data structure. For a single image, this did not prove costly in terms of memory as there were not many larger specular highlights present. At the end of the program, all the large specular highlight regions are removed by making a function call to the *Blob.Uncolor()* function. This function reverses the region growing process performed. On the other hand, to eliminate regions that that were smaller than a predetermined minimum size, the *Blob.Uncolor()* function was called in the iterative loops used in region growing. After this the image with the identified specular highlights is written as a binary image in JPG format.

3.3 Testing and Results

Testing of the program was primarily performed through brute force analysis. There were multiple input values into the program such as the Gaussian sigma, extrema neighborhood size etc. whose values were modified, and the expected results observed through a process of trial and error.

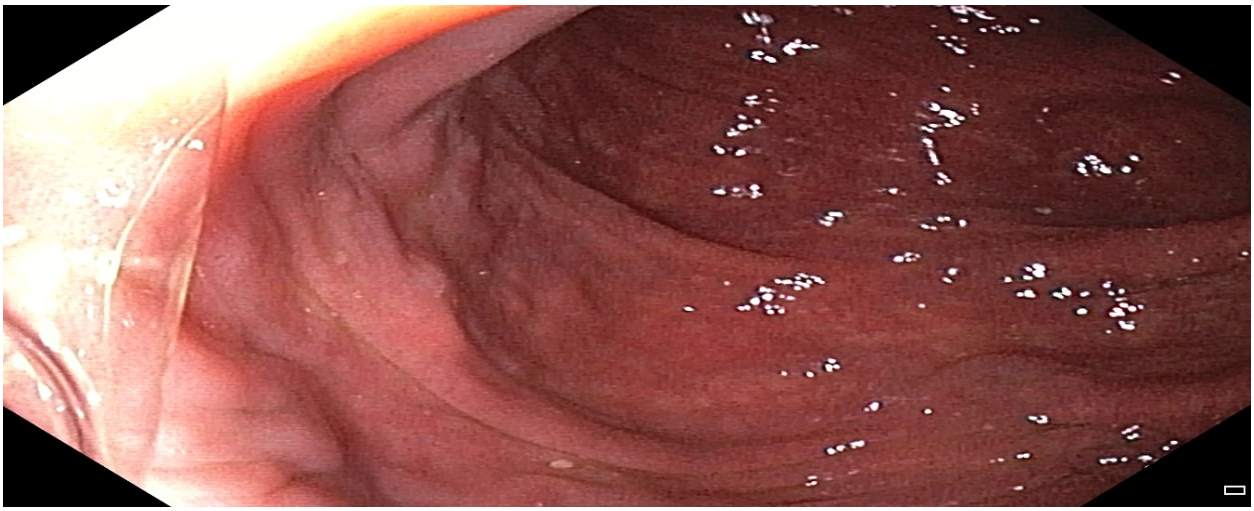


Figure 4: Test Image Used for Brute Force Analysis

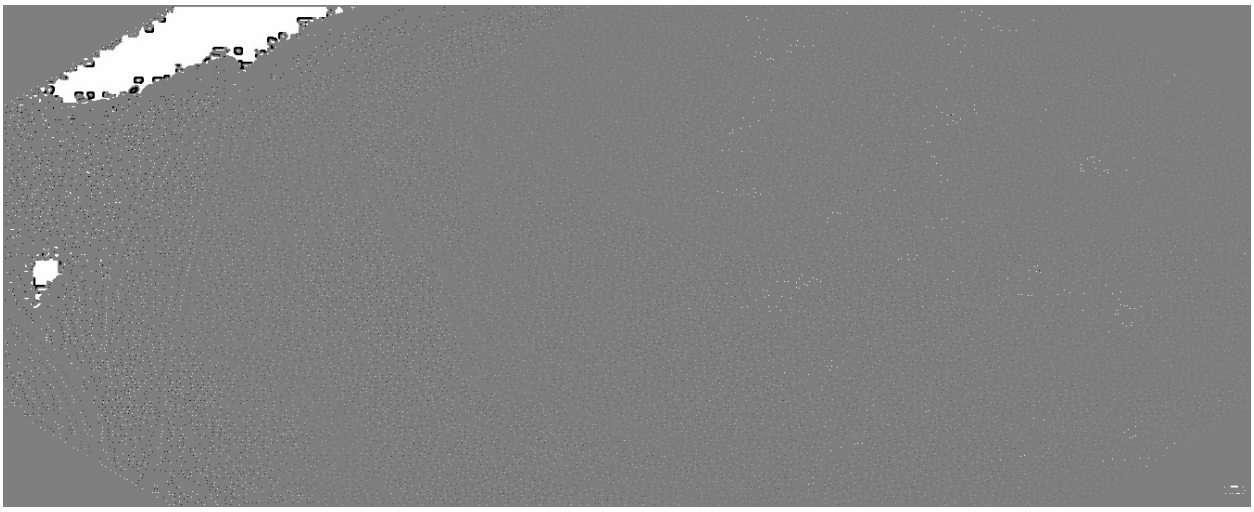


Figure 5: Extrema Performed on Test Image with Size 2

Figure 6: Gaussian Performed on Test Image with Sigma 0.25

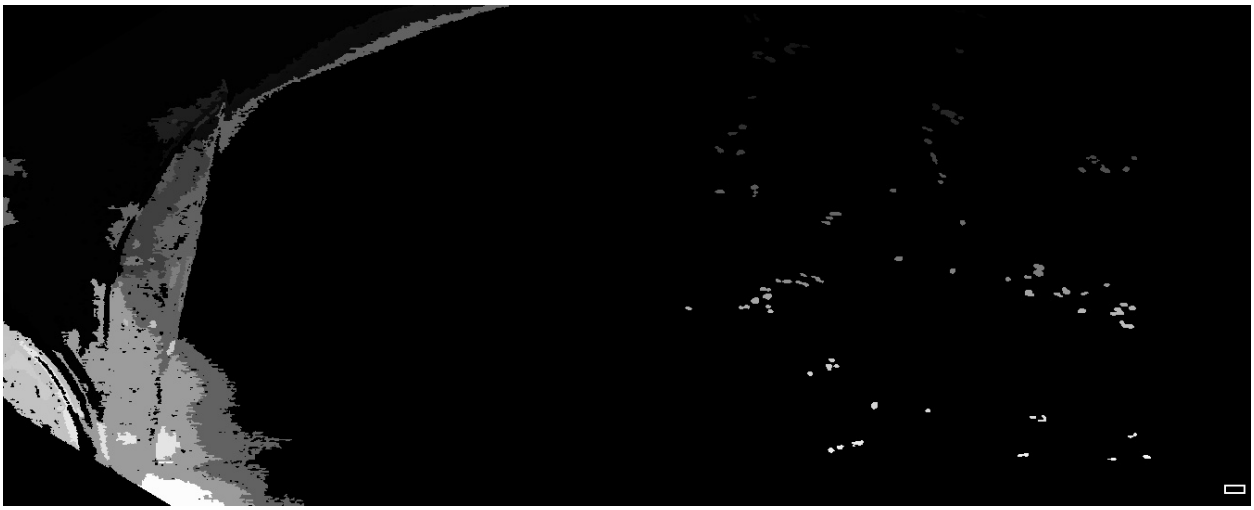


Figure 7: Region Growing Performed Using Gaussian And Extrema Images with Threshold Value of 225

Figure 8: Region Growing Performed Using Gaussian Sigma 0.8 And Threshold Value 235

Figure 9: Region Growing Performed Using Gaussian Sigma 0.9 And Threshold Value 235

Figure 10: Region Growing Performed Using Gaussian Sigma 1.0 And Threshold Value 235

Upon regulating the size of specular highlights, an improvement was seen in the results. However, it was observed that fine tuning the size of the specular highlights so that only all target specular highlights appear was challenging.

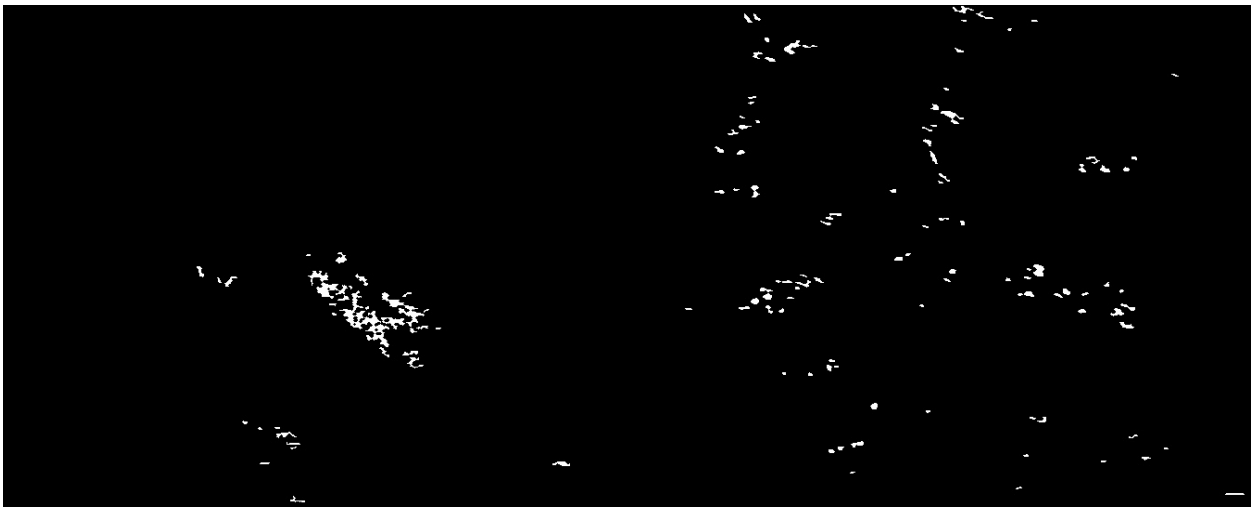


Figure 11: Region Growing Performed Using Maximum Size 1000 And Threshold Value 175

Figure 12: Region Growing Performed Using Maximum Size 500 And Threshold Value 175

On the other hand, adaptive thresholding was able to correctly identify the general location of specular highlights. Conversely, the downside included specular highlights whose original shapes were no longer preserved.

Figure 13: Region Growing Performed Using Median Size 3 And Threshold Value 100

Figure 14: Region Growing Performed Using Median Size 5 And Threshold Value 100

An important input to this program is the threshold value. The threshold value determines if a seed point will be selected hence altering whether a specular region will be grown or not. To better understand what values should be chosen for thresholding in a brute force study, an excel analysis was performed. To do so, pixel values that fell within a line across a specular highlight region were examined. For this starting and ending coordinates were chosen from the colonoscopy image in proximity to specular highlights. Then the pixel values were written to a comma separated value (csv) file. The graphs for the average, median, maximum and minimum pixel values were then examined.

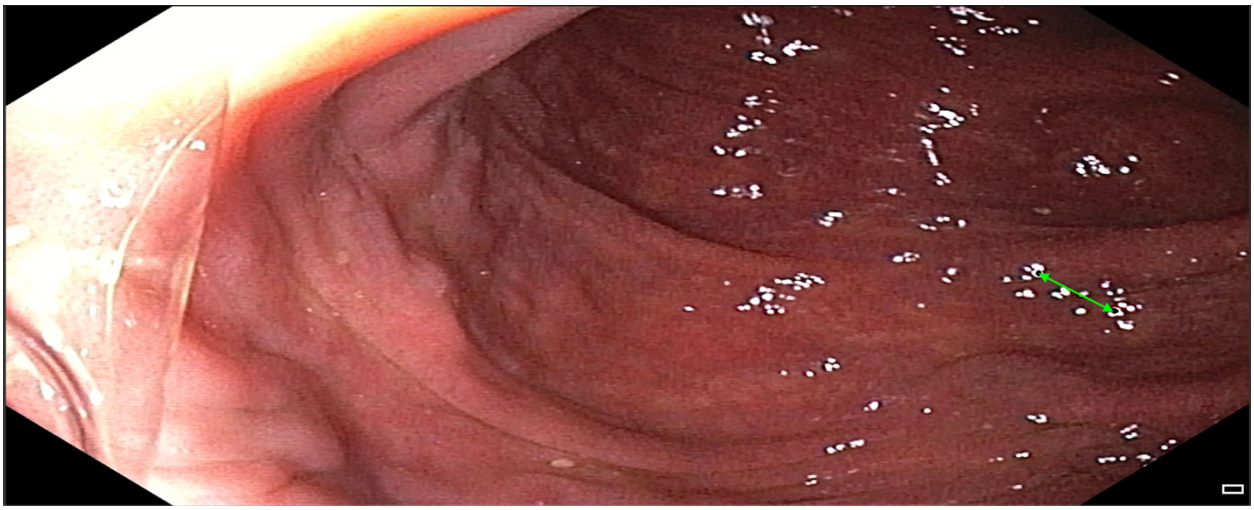
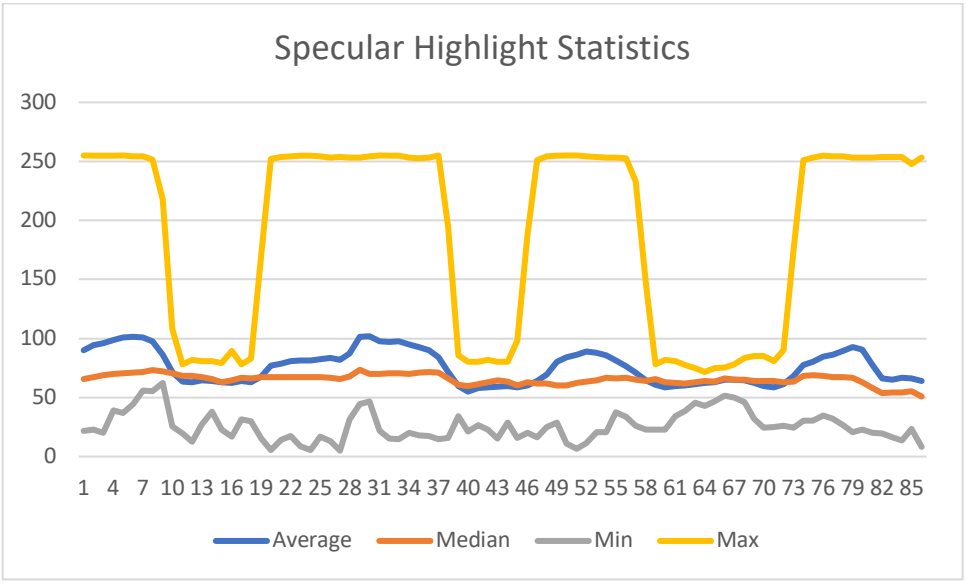


Figure 15: Double Arrow Line Indicates Sample Specular Region Examined



Graph # 1: Sample Specular Region Data

From the graph an understanding can be obtained about these specular highlight regions and how there brightness values alter. Closer to the peak in brightness (i.e. closer to the specular region), a slight dip in brightness can be observed. This dip is a dark halo that surrounds

the specular highlight on the colonoscopy image. It can be speculated that this dark halo may be a result of the colonoscope position and lighting.

All in all, region growing did not effectively identify all specular highlights. There were instances were false specular highlights were identified such as water bubbles present on the colon surface. On the other hand, there were also instances where some specular highlights were missing in the generated binary image as specular highlights of interest were removed. The reason for this was the inability to analyze an "ideal" threshold value that would ensure all true specular highlights are identified while ignoring all false specular highlights. In addition, as there were multiple parameters altering one either way would lead to dependencies on the other.

4. Using Background Subtraction to Find Specular Highlights

4.1 Algorithm Design

In colonoscopy images, the colon surface serves as the background. On the other hand, specular regions are outliers present on the image foreground. Therefore, subtracting the background from colonoscopy images would indicate the specular regions. This is termed as the process of background subtraction.

It is understood that a specular region will contain not only pixels with high brightness values (i.e. RGB values close to 255) but also will be comparatively brighter than their surrounding pixels. Thus, to perform background subtraction, the difference between each respective Gaussian and median pixel values is computed. This difference helps isolate the outliers. While computing the Gaussian image it is observed that all specular highlights on the original color image are smoothed. The reason behind this is that the Gaussian image determines a weighted average of all these pixel values in the image. As specular regions do not dominate

the original colonoscopy image, they do not carry much weight in the Gaussian calculation. Thus, they are not prominently present in the Gaussian image. On the other hand, the median image emphasizes the specular regions. It does so by providing information about surrounding pixels and hence highlighting the brightness difference between a specular pixel and its neighboring non-specular pixel values. The median calculation is performed with in a neighborhood size. Neighborhoods containing the specular highlight will determine a median value for the neighborhood that then amplifies the brightness value of the specular region. After this difference value is determined, an automatic threshold is performed to isolate the specular highlights from the image.

To further improve the program, double thresholding was also incorporated into the design. Double Thresholding added an additional check to determine whether the pixel under consideration was indeed a specular highlight on the foreground of the image. The two factors used to determine this were a threshold intensity value along with saturation-intensity values. Pixels below the threshold intensity values were rejected whereas those above had a possibility of being considered a specular highlight. Using the saturation-intensity values to distinguish non-specular regions from specular regions relied on the principle that specular pixels will have high intensity values but low saturation values. This is due to the high brightness value of a pixel in the specular region.

4.2 Implementation

Implementation of background subtraction requires the use of primarily two functions from the image processing library i.e. *Gaussian()* and *Median()* functions. After reading the color colonoscopy image, the Gaussian function is applied with an input sigma value. A median

function is also applied with a predetermined neighborhood size as the argument parameter. Subsequently, an output image is defined, and each pixel value is looped. This is done with the use of two nested for loops. Each pixel in the output image is assigned the difference value between the respective Gaussian and median pixel. Finally, a call to the *Threshold()* function is made given a threshold value as an argument parameter. Hence, the output image produced is a binary image containing the identified specular regions.

```
for (int y = 0; y < Input.Ydim; y++)
for (int x = 0; x < Input.Xdim; x++)
{
        Output.Data2D[y][x] = Gaussian.Data2D[y][x] - Median.Data2D[y][x];

}

im_float Thresh(Output);
Thresh.Threshold(threshold);
```

Figure 16: Background Subtraction Implementation

To improve the accuracy of specular highlight detection, double thresholding was implemented by first calculating the saturation-intensity values for the image. For this purpose, a separate program was written. This program took the RGB color colonoscopy image as an input. After this the function, *RGBToHSI(),*was called from the image processing library to calculate the hue, saturation and intensity values for the colonoscopy image. This function generates an output image where the R channel corresponds to hue values, G channel corresponds to the saturation values and the B channel corresponds to intensity values. Using these values, average saturation-intensity values are determined. The average values for both specular regions and non-specular regions are computed. To do so, the summation of saturation-intensity values are determined for both specular and non-specular regions. They are then divided by the number of specular and non-specular regions respectively. This is done by comparing the output image produced to a binary image where the specular regions have been manually identified.

```
for(int y = 0; y < TruthImage.Ydim; y++)
for(int x = 0; x < TruthImage.Xdim; x++)
{
        // Black
        if(TruthImage.Data2D[y][x] == 0)
        {
                totalFalseSaturationValues += Output.G.Data2D[y][x];
                totalFalseIntensityValues  += Output.B.Data2D[y][x];
                numberOfFalse++;
        }
        // White
        if(TruthImage.Data2D[y][x] == 1)
        {
                totalTrueSaturationValues += Output.G.Data2D[y][x];
                totalTrueIntensityValues  += Output.B.Data2D[y][x];
                numberOfTrue++;

        }
}

// Calculate average saturation and intensity values
float averageTrueSaturation = totalTrueSaturationValues / numberOfTrue;
float averageFalseSaturation = totalFalseSaturationValues / numberOfFalse;
float averageTrueIntensity = totalTrueIntensityValues / numberOfTrue;
float averageFalseIntensity = totalTrueIntensityValues / numberOfFalse;
```

Figure 17: Calculation of Average True Saturation-Intensity Values

Once these values are computed they are used to compute a midpoint saturation-intensity values as follows:

$$Midpoint\ Saturation = (averageTrueSaturation + averageFalseSaturation)/2$$

$$Midpoint\ Intensity = (averageTrueIntensity + averageFalseIntensity)/2$$

For double thresholding, each pixel value is compared to a predefined threshold value. In addition, it is ensured that each pixel value has a saturation value less than the constant saturation parameter and an intensity value greater than the constant intensity parameter. The average true saturation-intensity values along with the midpoint-saturation values are used as the constant parameter values. To threshold each pixel the *RGBToHSI()* function is called once again. Subsequently, the computed saturation-intensity values are then compared to the constant saturation-intensity parameters. If all three conditions are satisfied, then it can be deduced that

the pixel belongs in a specular highlight region. Therefore, the pixel value on the output image is set to the difference value between the computed Gaussian and median values. On the contrary, if all three conditions are not satisfied it means that the pixel value is not a specular pixel.

```
for (int y = 0; y < Input.Ydim; y++)
for (int x = 0; x < Input.Xdim; x++)
{
     if(Gaussian.Data2D[y][x] - Median.Data2D[y][x] >= Threshold)
     {
             //Output.Data2D[y][x] = Gaussian.Data2D[y][x] - Median.Data2D[y][x];

        if ((HSI.G.Data2D[y][x] <= AverageSaturation) &&
            (HSI.B.Data2D[y][x] >= AverageIntensity))
        {
             Debug.Data2D[y][x] = 1;
             count_true++;
             Output.Data2D[y][x] = Gaussian.Data2D[y][x] - Median.Data2D[y][x];
        }
        else
        {
             Debug.Data2D[y][x] = -1; count_false++;
        }
     }
}
Debug.WriteJpg("debug.jpg");
cout << "Count True: " << count_true << endl;
cout << "Count False: " << count_false << endl;
```

Figure 18: Double Thresholding Implementation

4.3 Testing and Results

Testing the background subtraction method for accuracy involves choosing optimal parameters for the threshold value, Gaussian sigma and median neighborhood size. Optimal parameters would be defined as parameters that identify the most specular highlights from the original colonoscopy image. For this purpose, it is essential to have a means to compare your generated image from test parameters with an image that contains correctly pre-identified specular highlights. To achieve this, an OpenGL interactive paint program written by Dr. Gauch was used. This paint program would allow the user to manually identify specular highlights on a color image. It did so by allowing the user to place their cursor over a specular region to color it.

Additional functionality included removing color from an incorrectly identified pixel and altering the brush size (i.e. the number of pixels whose color would be changed when the cursor is placed over). The gradual result of this program was a binary image with the correctly identified specular regions of interest from the original image. This will be referred to as the "truth" image.

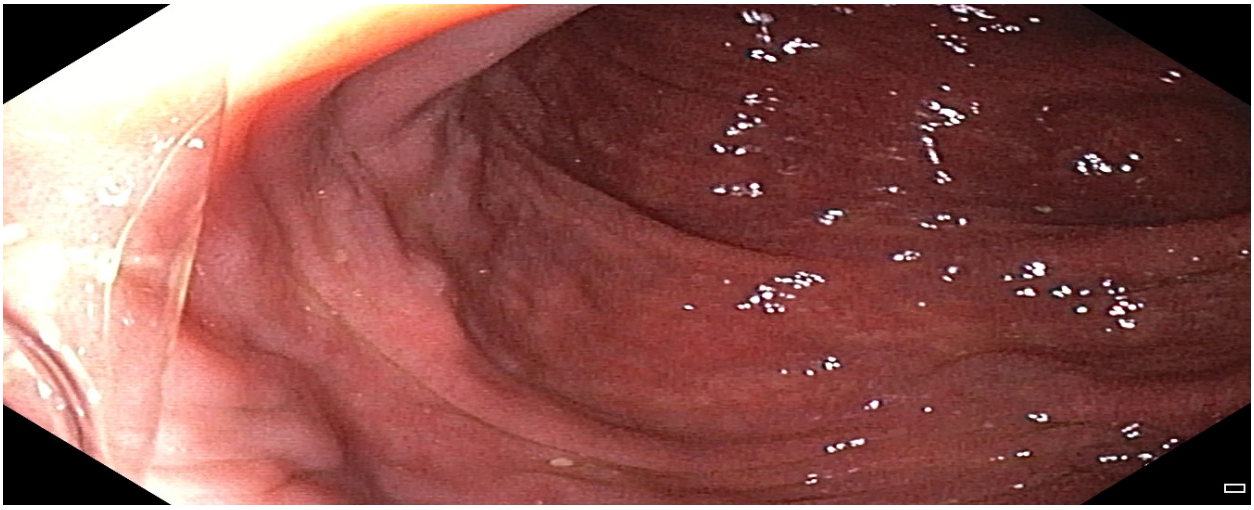


Figure 19: Test Image 1

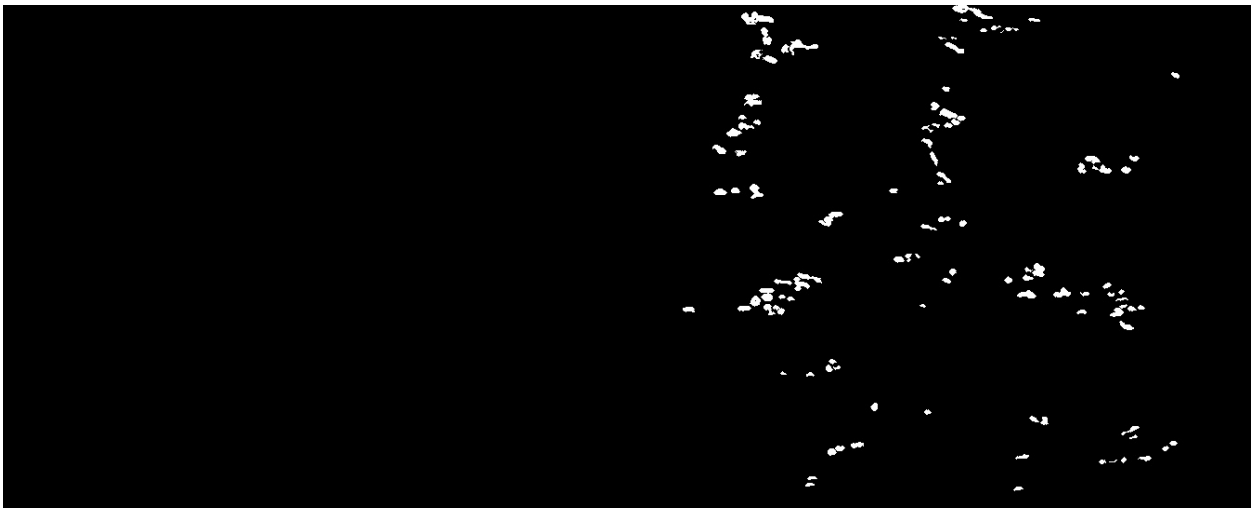


Figure 20: Truth Image for Test Image 1

The next step would be to implement a function that could compare the truth image to the test image generated from using background subtraction (i.e. the "experiment" image). This comparison would require each pixel from both images to be examined in parallel. For this purpose, the pixel value for the experiment image will be subtracted from the truth image pixel value. As the images are binary, the pixels contain two possible values i.e. zero for black and one for white. Therefore, the difference result helps determine four possible outcomes i.e. true negatives, false negatives, false positives, and true positives.

Experiment Image Pixel Value

Truth Image Pixel Value

Subtract	0	1
0	0 : *True Negative* i.e. non-specular region correctly identified in experiment image	-1 : *False Positive* i.e. specular region incorrectly identified in experiment image
1	1 : *False Negative* i.e. non-specular region incorrectly identified in experiment image	0: *True Positive* i.e. specular region correctly identified in the experiment image

Table 1: Truth Image and Experiment Image Difference Results

```
void compareTruth(im_float& truth, im_float& experiment,
                  int& truePos, int& falsePos, int& trueNeg, int& falseNeg)
{
        for(int y = 0; y < truth.Ydim; y++)
        for(int x = 0; x < truth.Xdim; x++)
        {
                int result = 0;
                result = truth.Data2D[y][x] - experiment.Data2D[y][x];

                // Specular highlight does not exist in truth image nor in the experiment image
                if(result == 0 && truth.Data2D[y][x] == 0)
                        trueNeg++;
                // Missing Specular Highlight in Experiment Image
                else if(result == 1)
                        falseNeg++;
                // Specular highlight present in truth image and the experiment image
                else if(result == 0 && truth.Data2D[y][x] == 1)
                        truePos++;
                // Extra Specular Highlight in Experiment Image
                else if(result == -1)
                        falsePos++;
        }
}
```

Figure 21: Function Implementation to Compare Truth and Experiment Image

Using the above information variables, a percent error can be computed. The percent error would be based on the number of incorrectly identified regions over the total number of regions. Therefore, we are looking for the test image with the lowest percent error to correctly identify the optimal parameters.

```
    // Calculate % error
    int denominatorSum = numberOfTruePositives + numberOfTrueNegatives + numberOfFalsePositives + numberOfFalseNegatives;

    float percentError = (float) (numberOfFalsePositives + numberOfFalseNegatives) / (float) (denominatorSum);
```

Figure 22: Percent Error Computation

To find the image with the lowest percent error, experiments were run using a range of values for the three parameters. To allow testing given range of values three nested for loops were incorporated into the background subtraction program. The loops iterated from the start range of the parameter to its end range.

```
for (float sigma = StartSigmaRange; sigma <= EndSigmaRange; sigma += 0.1)
{
     // Perform gaussian filter
     im_float Gaussian(Input);
     Gaussian.Gaussian(sigma);
     Gaussian.WriteJpg((char *)"gaussian.jpg");

     for (int size = StartMedianSizeRange; size <= EndMedianSizeRange; size += 1)
     {
             // Apply Median Filter
             im_float Median(Input);
             Median.Median(size,size);
             Median.WriteJpg((char *) "median.jpg");

             for (int threshold = StartThresholdRange; threshold <= EndThresholdRange; threshold += 1)
             {
                     for (int y = 0; y < Input.Ydim; y++)
                     for (int x = 0; x < Input.Xdim; x++)
                     {
                             Output.Data2D[y][x] = Gaussian.Data2D[y][x] - Median.Data2D[y][x];

                     }
```

Figure 23: Testing Range of Parameter Values

The experiments conducted consisted of performing a 2D brute force search in the following manner:

1) Run the program by keeping two parameters constant and change the remaining parameter over time by entering a range value.

2) From the results obtained, choose the one with the most resemblance to the original image (i.e. the lowest percent error). This is the optimal value obtained at the moment.

3) Run the experiment program by narrowing the range of the parameter in question and/or refining the step size for loop iteration. Choose a narrow range that includes the optimal value obtained from the previous step.

4) Continue to repeat steps 2 and 3 until you have the most optimal value attainable.

5) Repeat the same procedure for the other two parameters.

The above 2D brute force algorithm was performed on the test image. The first parameter to be modified was the threshold value while the Gaussian sigma and median size was kept constant at 0.6 and 12 respectively. The ideal threshold value determined was 75.

```
Conclusion:
Image with Lowest Percent Error: thresh0.600000 12 75.jpg
Lowest Percent Error: 0.0052414
```

Figure 24: Output Produced Upon Varying Threshold Value

Figure 25: Image Produced with Optimal Threshold Value

Next the median neighborhood size was modified while the Gaussian sigma and threshold value was kept constant at 0.6 and 79 respectively. The ideal median size determined was 29.

```
Conclusion:
Image with Lowest Percent Error: thresh0.600000 29 75.jpg
Lowest Percent Error: 0.00414875
```

Figure 26: Output Produced Upon Varying Median Value

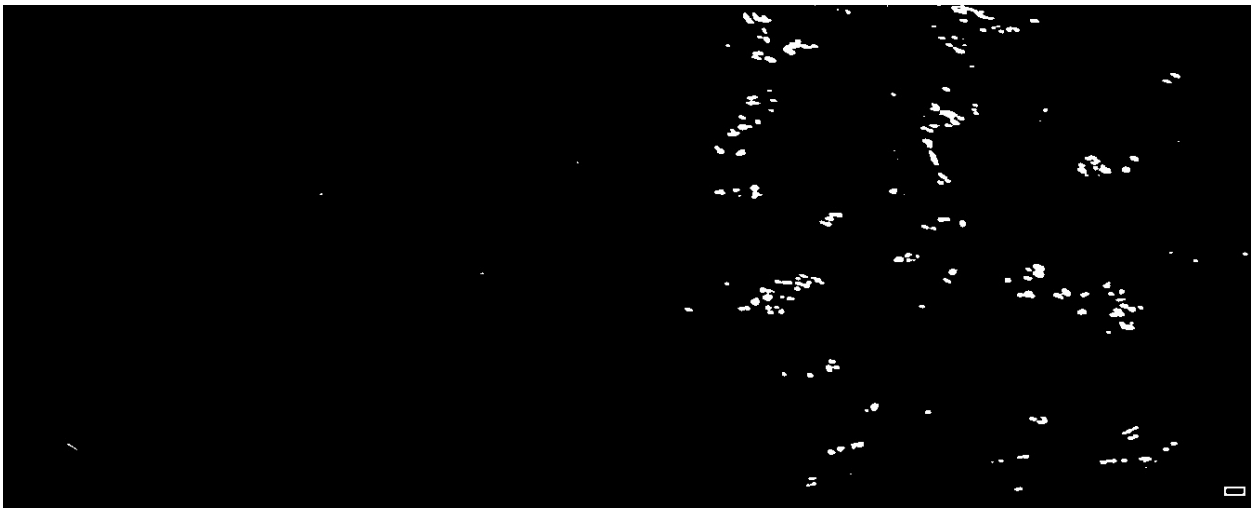


Figure 27: Image Produced with Optimal Threshold and Median Value

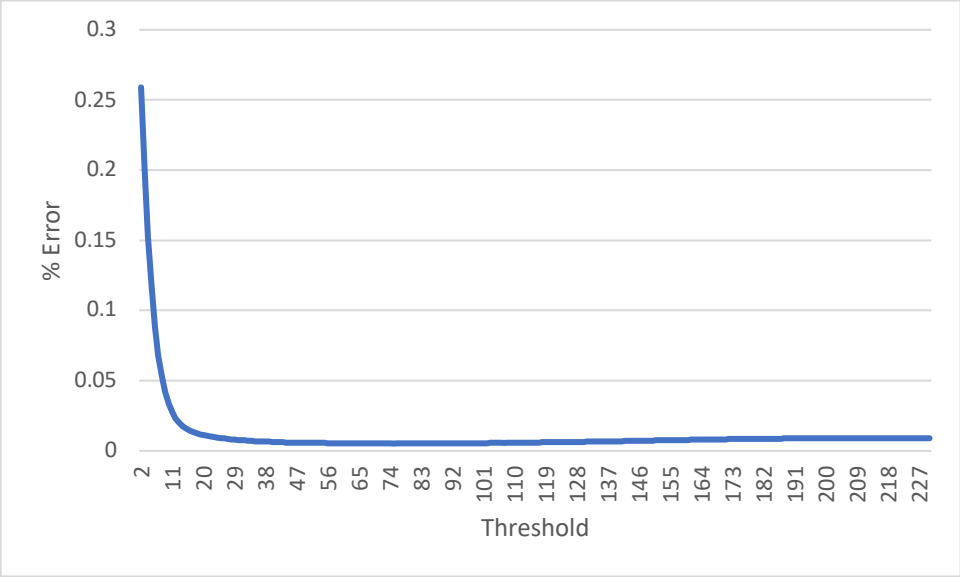
Lastly the sigma value was modified over a range. The final optimal parameters determined were 1.0 for the Gaussian sigma, 29 for the median neighborhood size and 75 for the threshold value.

```
Conclusion:
Image with Lowest Percent Error: thresh1.000000 29 75.jpg
Lowest Percent Error: 0.00403853
```

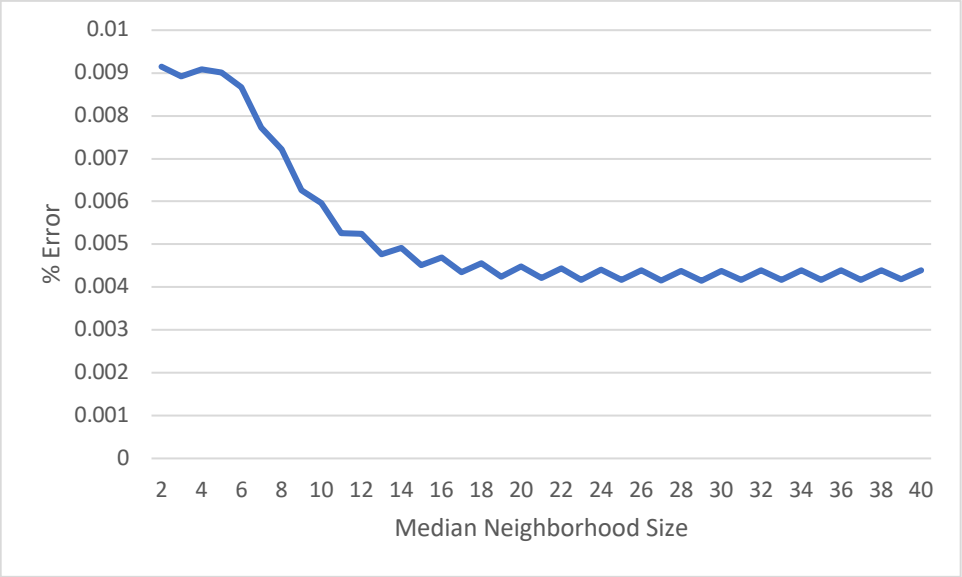
Figure 28: Output Produced Upon Varying Sigma Value

Figure 29: Image Produced with Optimal Values

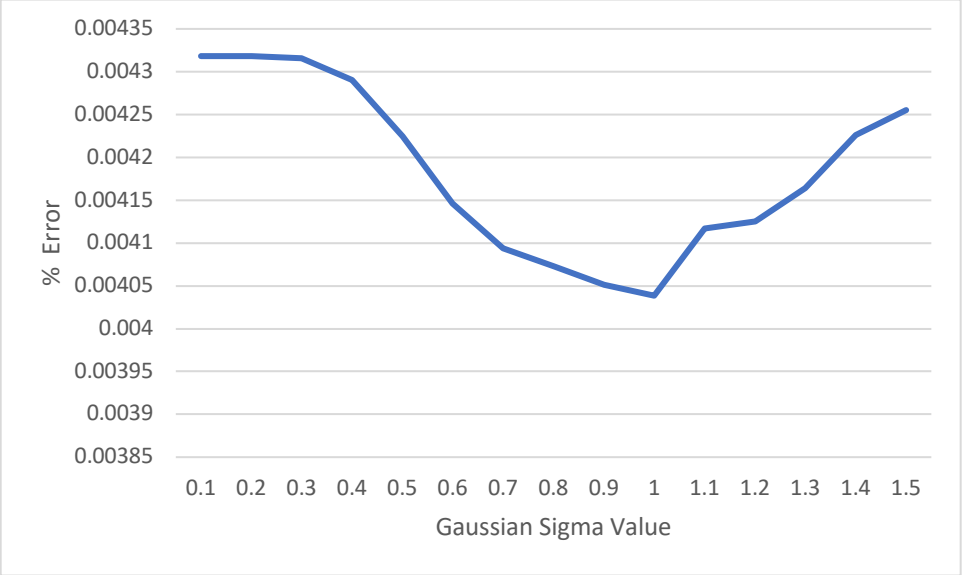
These results were then analyzed using excel to gain a better understanding. The graph for the threshold value and Gaussian sigma reaches a low range where the optimal value presides. However, the median value graph fluctuated towards the end of the plotted graph.



Graph 2: Change in Threshold Value and Its Effect on Percent Error



Graph 3: Change in Median Neighborhood Value and Its Effect on Percent Error



Graph 4: Change in Gaussian Sigma Value and Its Effect on Percent Error

The optimal parameters found for the above image were then applied on a few more images. This was done to discern whether parameters could potentially be generalized or not. It was observed that the percent errors obtained were relatively low and narrow in their difference. In addition, the mean and median values of parameters were also applied. This also yielded fairly

low percent errors. However, this observation may not be considered as conclusive as the parameters were not applied to a large number of colonoscopy images with varying image quality.

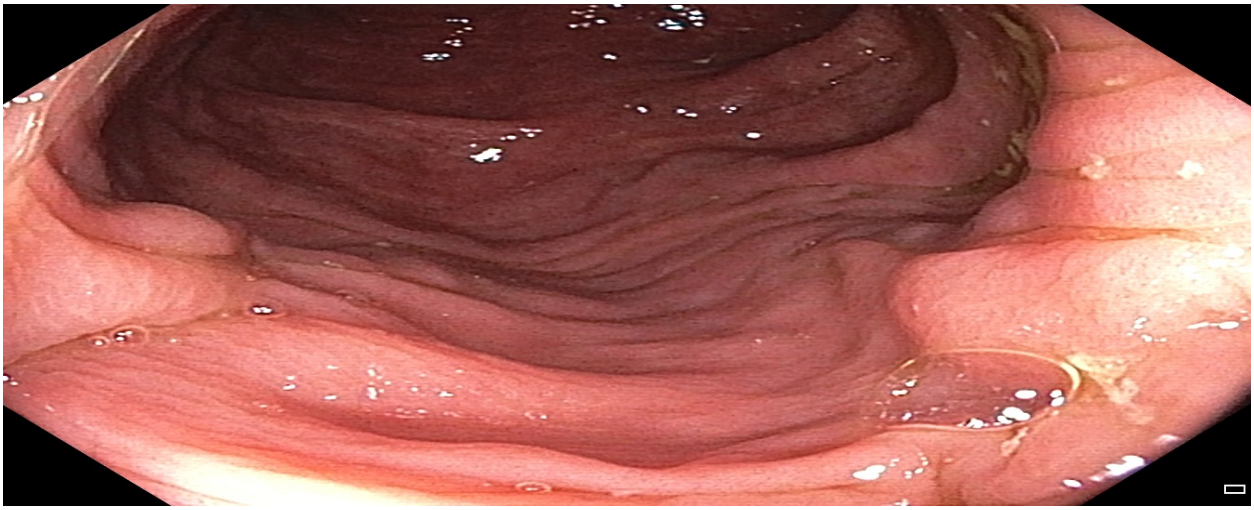


Figure 30: Test Image 2

Parameters	True -ve	True +ve	False -ve	False +ve	TN+TP+FN+FP	Image Size	% Error
thresh1.000000 29 75.jpg	721353	2001	1534	872	725760	725760	0.00331515

Conclusion:
Image with Lowest Percent Error: thresh1.000000 29 75.jpg
Lowest Percent Error: 0.00331515

Figure 31: Output Produced When Optimal Parameters Are Applied to Test Image 2

Figure 32: Image Produced When Optimal Parameters Are Applied to Test Image 2

Figure 33: Test Image 3

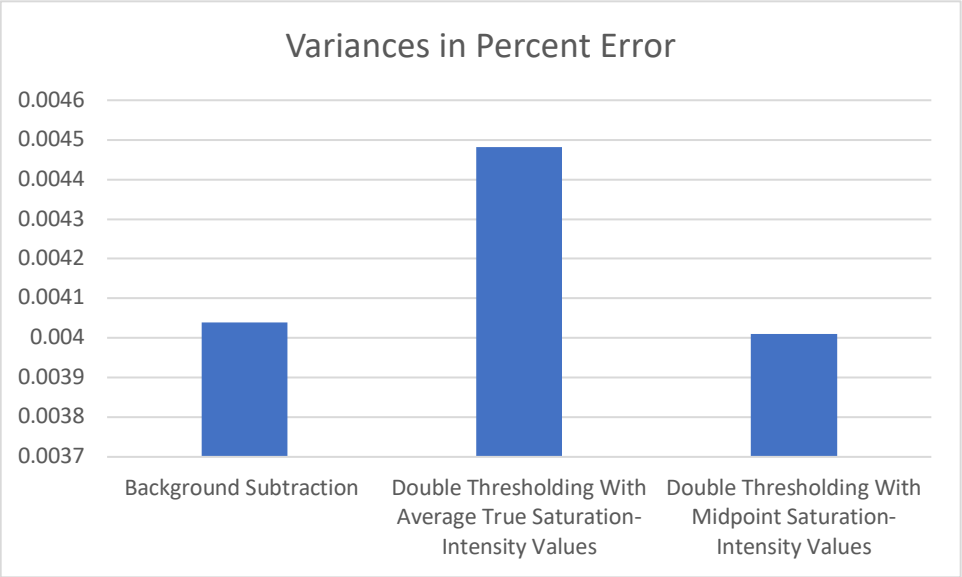
Parameters	True -ve	True +ve	False -ve	False +ve	TN+TP+FN+FP	Image Size	% Error
thresh1.000000 29 75.jpg	721799	2436	591	934	725760	725760	0.00210125

Conclusion:
Image with Lowest Percent Error: thresh1.000000 29 75.jpg
Lowest Percent Error: 0.00210125

Figure 34: Output Produced When Optimal Parameters Are Applied to Test Image 3

Figure 35: Image Produced When Optimal Parameters Are Applied to Test Image 3

Next tests performed involved examining the use of background subtraction with double thresholding. To perform these tests, we used the midpoint and average values for true saturation-intensity in thresholding. Using the midpoint as parameter values for saturation-intensity led to some of the specular highlights being removed. On the other hand, using the average true saturation-intensity values as parameters led to the opposite effect i.e. many non-specular regions being identified as specular highlights. Therefore, it was concluded that this technique was not entirely useful as no drastic decrease in percent error was observed.



Graph 5: Performance of Background Subtraction vs. Background Subtraction and Double Thresholding

5. A Brief Look into Image Restoration

5.1 Algorithm Design

Image restoration for colonoscopy images includes removal of the specular highlights from the image. A simple approach to achieve this would be to replace the specular highlight

pixels. These pixels may be replaced with pixel values that are similar in nature to the adjacent non-specular pixels. For this purpose, the Gaussian value may be computed and used as the replacement pixel. In this case, non-specular pixels will be unchanged (i.e. have the same value as the original image) whereas the Gaussian value will be substituted for the identified specular pixels.

5.2 Implementation

To implement the restoration program, the original colonoscopy color image and the "segmented" image (i.e. the image containing manually identified specular highlights) are required as inputs.

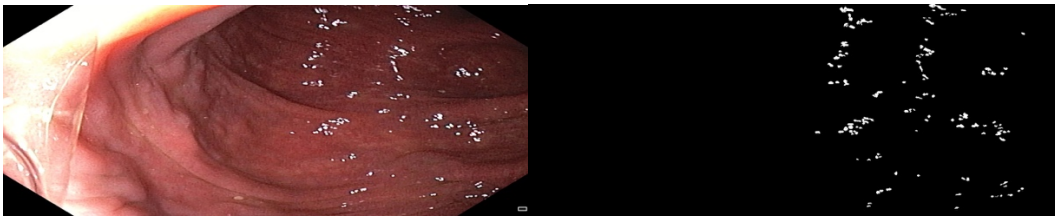


Figure 36 and 37: Original Color Colonoscopy Image and The Corresponding Segmented Image

Additionally, an input for the sigma value is obtained. This value is then sent as an argument to the *Gaussian()* function called from the image processing library. Subsequently, an output image is declared with the same dimensions as the input image. This output image is a color image and therefore has three two dimensional arrays analogous to the R,G and B color values.

Two nested for loops are used to iterate through the segmented image. In the case where the segmented pixel value is zero (i.e. a non-specular pixel), the corresponding output pixels are assigned the values from the original color image. On the other hand, if the segmented pixel

value is one (i.e. a specular pixel), then the corresponding output pixels are assigned the Gaussian values.

```
for (int y = 0; y < InputImg.Ydim; y++)
for (int x = 0; x < InputImg.Xdim; x++)
{
        if(Segment.Data2D[y][x] == 0)
        {
                Output.R.Data2D[y][x] = Input.R.Data2D[y][x];
                Output.G.Data2D[y][x] = Input.G.Data2D[y][x];
                Output.B.Data2D[y][x] = Input.B.Data2D[y][x];
        }
        if(Segment.Data2D[y][x] == 1)
        {
                Output.R.Data2D[y][x] = Gaussian.R.Data2D[y][x];
                Output.G.Data2D[y][x] = Gaussian.G.Data2D[y][x];
                Output.B.Data2D[y][x] = Gaussian.B.Data2D[y][x];
        }
}
```

Figure 38: Implementation for Image Restoration

5.3 Testing and Results

The program was tested with a range of sigma values. Despite the identified specular regions being replaced, it was observed that the dark halo surrounding the specular region still remained.

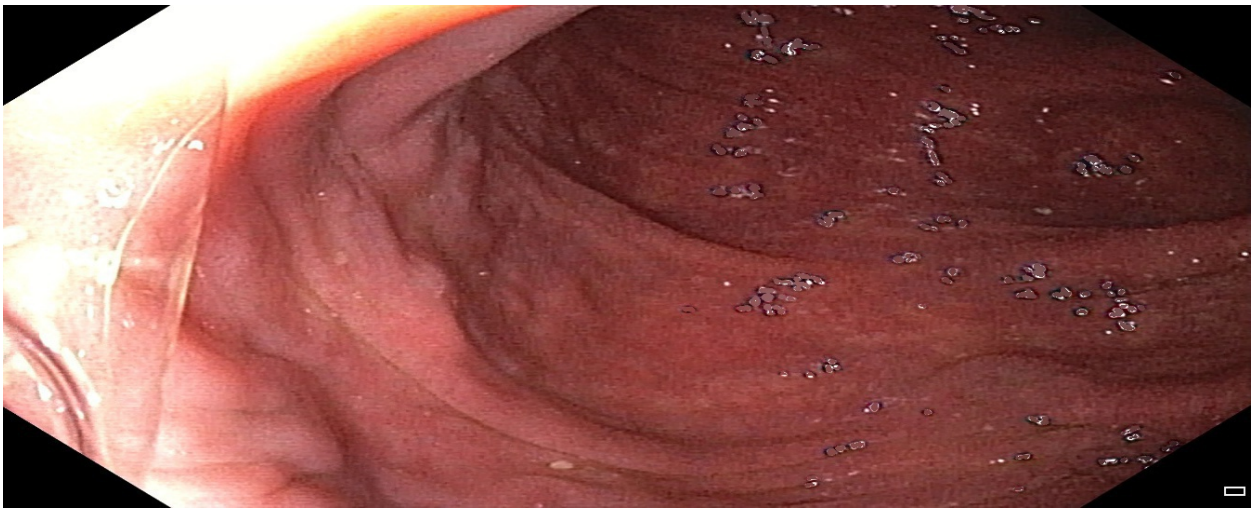


Figure 39: Image Represents the Dark Halo Present Around the Specular Region

To remove this dark circle the *Dilate()* function was called on the segmented image from the image processing library. This function eliminates the dark halo by expanding the specular region on the segmented image. The amount of expansion is controlled by a radius value sent as an argument parameter.

After this a range of sigma and radius values were tested and the resulting images examined.

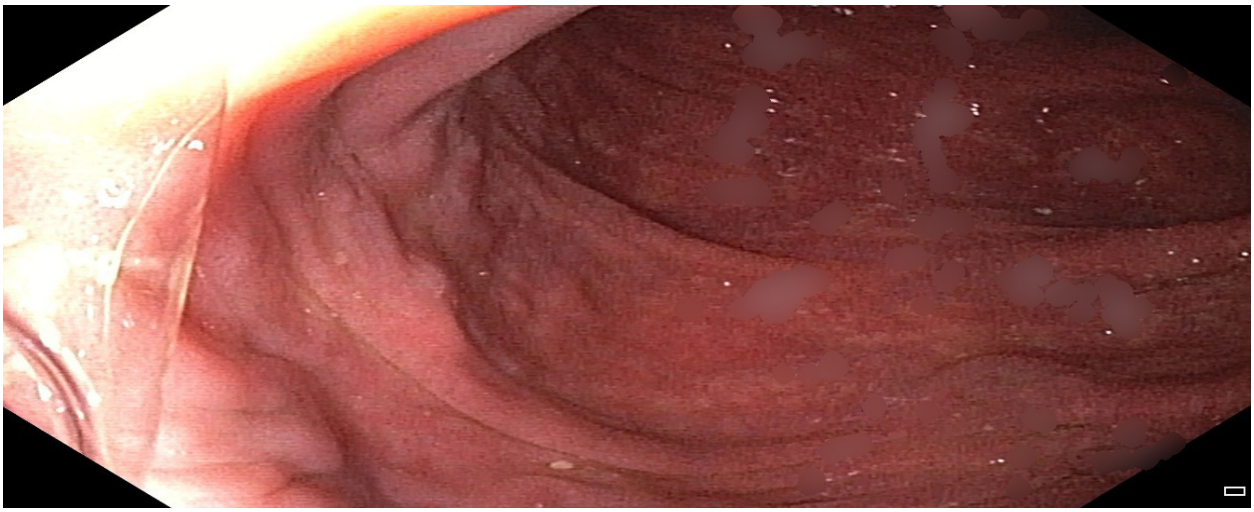


Figure 40: Restored Image with Sigma Value 4 And Radius Value 20

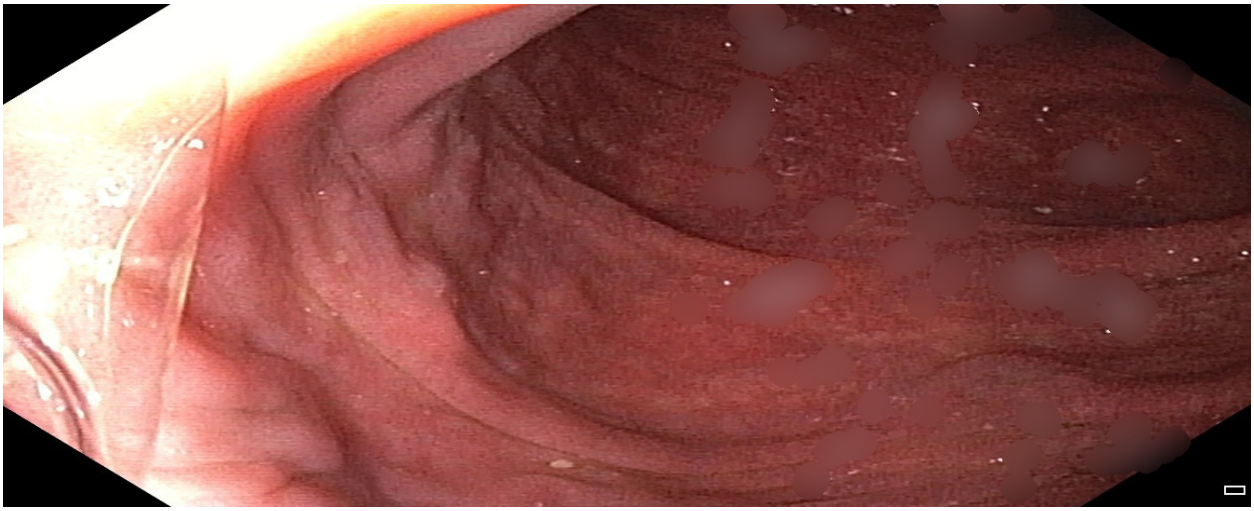


Figure 41: Restored Image with Sigma Value 5 And Radius Value 19

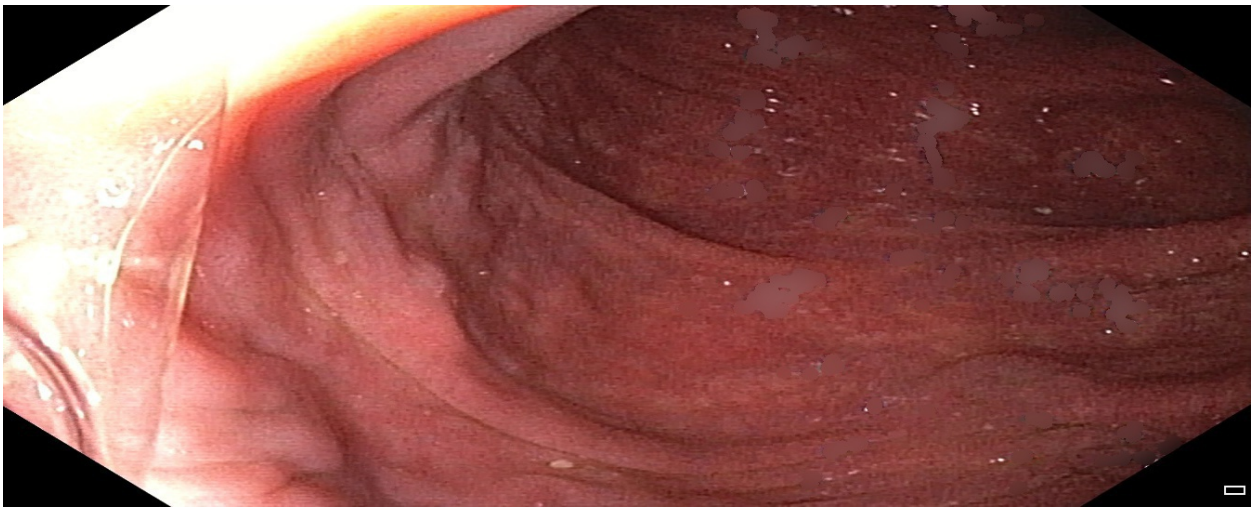


Figure 42: Restored Image with Sigma Value 5 And Radius Value 20

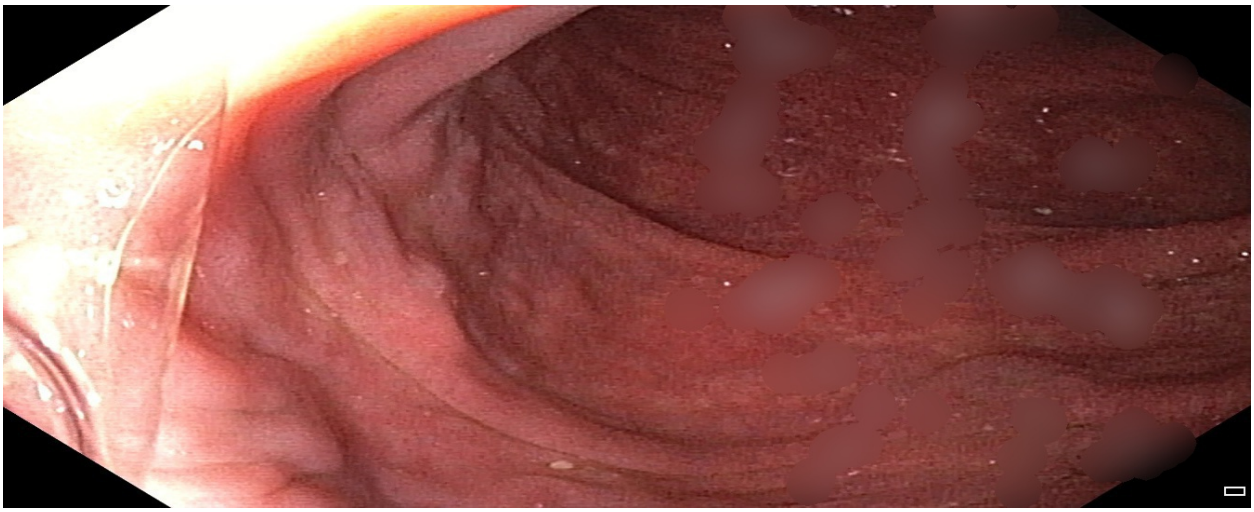


Figure 43: Restored Image with Sigma Value 6 And Radius Value 20

As observed from the above images, the Gaussian replacement does not blend in with the original image. The Gaussian replacement does not contain the texture or contour of the colon structure and is too smooth. Hence, it may be concluded that further improved methods may be required to restore colonoscopy images.

6. Conclusions

Specular highlights that occur on colonoscopy images can lead to a loss of information for doctors. The purpose of this research project was to identify specular highlights formed on colonoscopy images due to the colonoscope angle. Two primary methods were relied upon for specular highlight identification i.e. region growing and background subtraction.

Region growing involved selection of a seed point followed by accumulation of similar neighboring pixels. Seed points were selected using thresholding based off of an extrema image. In addition, later modifications were made to use adaptive thresholding which relied on both extrema and median images. With the thresholding approach it was observed that not all specular

highlights were identified correctly. There were instances when non-specular highlights were perceived as specular highlights and vice-versa. However, in adaptive thresholding the specular highlights were identified but without preservation of their original shape. The second method, background subtraction, identifies the specular highlights as outliers on the foreground of the image. It does this by using the net effect of the Gaussian and median calculation of the colonoscopy image. This approach was later combined with double thresholding using saturation-intensity values. Doing so did not add drastic improvements to the specular region identification process.

Amongst the two methods background subtraction seemed the better performing. This is because it was for the most part able to identify specular highlights correctly alongside retaining their silhouette on the image. Moreover, background subtraction was less reliant on a larger number of parameters compared to region growing.

The research project also examined a simple image restoration technique. This technique restored the specular region by replacing the pixel with Gaussian smoothing values. This approach yielded results that can be massively improved on by utilizing more complex image restoration techniques.

7. Future Work

The image segmentation techniques used in this paper were mainly tested by being applied on a single image. Future steps may include determining whether the same parameters can be applied for a series of thousands of images. It will also include examining whether

additional generalizations need to be made such as duller images have certain parameters in comparison to more brighter images.

In addition, to restore the image, multiple techniques can be examined in the future. A potential approach may be to copy and paste a similar non-specular area onto the specular region. The specular region can also be replaced with median values instead of Gaussian values. Additional complexity can be added to this method by using spatial and temporal median values for replacement. This would require examination of median values as they differ over time and space in a sequence of colonoscopy images.

Though this paper examines image segmentation methods and attempts a simple approach at image restoration, it is not conclusive and encourages further research in the area. Therefore, additional techniques or a combination of techniques can be experimented with that may be better both in accuracy and efficiency.

8. References

[1] M. Maciejewski, W. Surtel and T. Malecka-Massalska, "Level-set image processing methods in medical image segmentation," *2012 Joint Conference New Trends In Audio & Video And Signal Processing: Algorithms, Architectures, Arrangements And Applications (NTAV/SPA)*, Lodz, 2012, pp. 39-41.

[2] Pham, Dzung L., Chenyang Xu, and Jerry L. Prince. "Current methods in medical image segmentation." Annual review of biomedical engineering 2.1 (2000): 315-337.

[3] P. K. Jain and S. Susan, "An adaptive single seed based region growing algorithm for color image segmentation," 2013 Annual IEEE India Conference (INDICON), Mumbai, 2013, pp. 1-6.

[4] W. Deng, W. Xiao, H. Deng and J. Liu, "MRI brain tumor segmentation with region growing method based on the gradients and variances along and inside of the boundary curve," 2010 3rd International Conference on Biomedical Engineering and Informatics, Yantai, 2010, pp. 393-396.

[5] S. S. Mohamed, N. M. Tahir and R. Adnan, "Background modelling and background subtraction performance for object detection," 2010 6th International Colloquium on Signal Processing & its Applications, Mallaca City, 2010, pp. 1-6.

[6] Aswathi, V. M., and James Mathew. "A review on image restoration in medical images." *Compusoft* 4.4 (2015): 1588.

[7] S. Bardhan, S. Nath and M. K. Bhowmik, "Evaluation of background subtraction effect on classification and segmentation of knee thermogram," 2017 8th International Conference on Computing, Communication and Networking Technologies (ICCCNT), Delhi, 2017, pp. 1-7.