INDIAN INSTITUTE OF TECHNOLOGY

KHARAGPUR

DEPARTMENT OF ELECTRONICS AND ELECTRICAL COMMUNICATION

ASSIGNMENT NUMBER 2

ARNAB BISWAS

21EC65R01

VISUAL INFORMATION PROCESSING AND EMBEDDED SYSTEMS

M.Tech.

2021 – 2022

Introduction:

Histogram equalization:

This method usually increases the global contrast of many images, especially when the usable data of the image is represented by close contrast values. Through this adjustment, the intensities can be better distributed on the histogram. This allows for areas of lower local contrast to gain a higher contrast. Histogram equalization accomplishes this by effectively spreading out the most frequent intensity values. The method is useful in images with backgrounds and foregrounds that are both bright or both dark. In particular, the method can lead to better views of bone structure in x-ray images, and to better detail in photographs that are over or under-exposed. A key advantage of the method is that it is a fairly straightforward technique and an invertible operator. So, in theory, if the histogram equalization function is known, then the original histogram can be recovered. The calculation is not computationally intensive. A disadvantage of the method is that it is indiscriminate. It may increase the contrast of background noise, while decreasing the usable signal.

Histogram Matching:

In image processing, histogram matching or histogram specification is the transformation of an image so that its histogram matches a specified histogram. The well-known histogram equalization method is a special case in which the specified histogram is uniformly distributed. It is possible to use histogram matching to balance detector responses as a relative detector calibration technique. It can be used to normalize two images, when the images were acquired at the same local illumination (such as shadows) over the same location, but by different sensors, atmospheric conditions or global illumination.

Algorithm:

I have used many functions to draw the histogram equalized image and the histogram matched images.

1) `create_hist`: This function is used to create the count of the pixels of the image. The function returns a vector where the count of each pixel value is stored.

```
vector<int>create_hist(Mat img)
{
    vector<int>hist;
    hist.resize(256, 0);
    for (int i = 0; i < img.rows; i++)
    {
        for (int j = 0; j < img.cols; j++)
        {
            int val = (int)img.at<uchar>(i, j);
            hist[val]++;
        }
    }
    return hist;
}
```

2) `normalhist`: This function is used to get the probability distribution function of the image. The function returns a vector where the normalized histogram value is stored. In this function, the count of each pixel is divided by the size of the image.

```
vector<double>normalhist(vector<int>hist)
{
    vector<double>nor_hist;
    double val = 256.0 * 256.0;
    nor_hist.resize(256, 0);
    for (int i = 0; i < 256; i++)
    {
        nor_hist[i] = hist[i] / val;
    }
    //for (auto x : nor_hist) cout << x << " ";
    return nor_hist;
}
```

3) `sum_all`: This function is used to find the cumulative distribution function of the image and then find the histogram equalized array. Here, the used formula is

$$S_k = (L-1)cdf(x)$$

, where L means the number of possible image.

```
vector<int>sum_all(vector<double>nor_hist)
{
    vector<int>last;
    vector<double>last1;
    last1.resize(256, 0);
    last.resize(256, 0);
    last1[0] = nor_hist[0];
```

```
        for (int i = 1; i < 256; i++)
        {
            last1[i] = nor_hist[i] + last1[i - 1];
        }

        for (int i = 0; i < 256; i++)
        {
            last1[i] = last1[i] * 255;
        }
        for (int i = 0; i < 256; i++)
        {
            last[i] = floor(last1[i]);
        }
        //for (auto x : last) cout << x << " ";
        return last;
    }
```

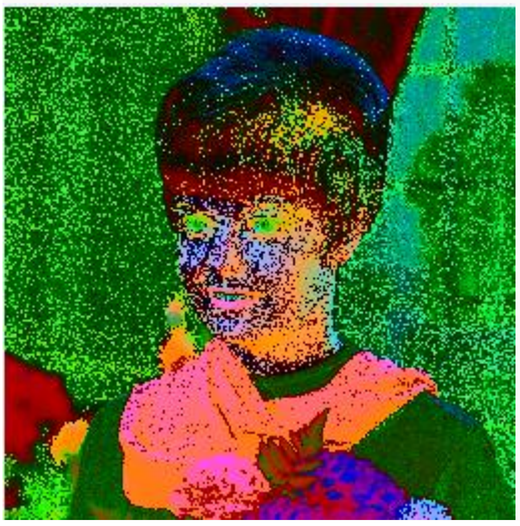
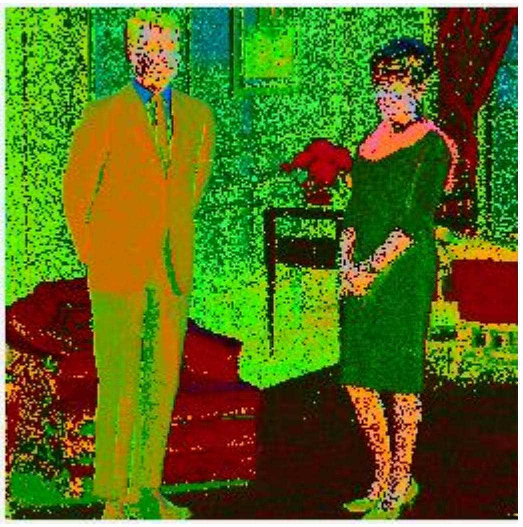
4) `showHistogram:` This function is used to show the histogram of the images on the console. I have used the line and point function to create the histogram.

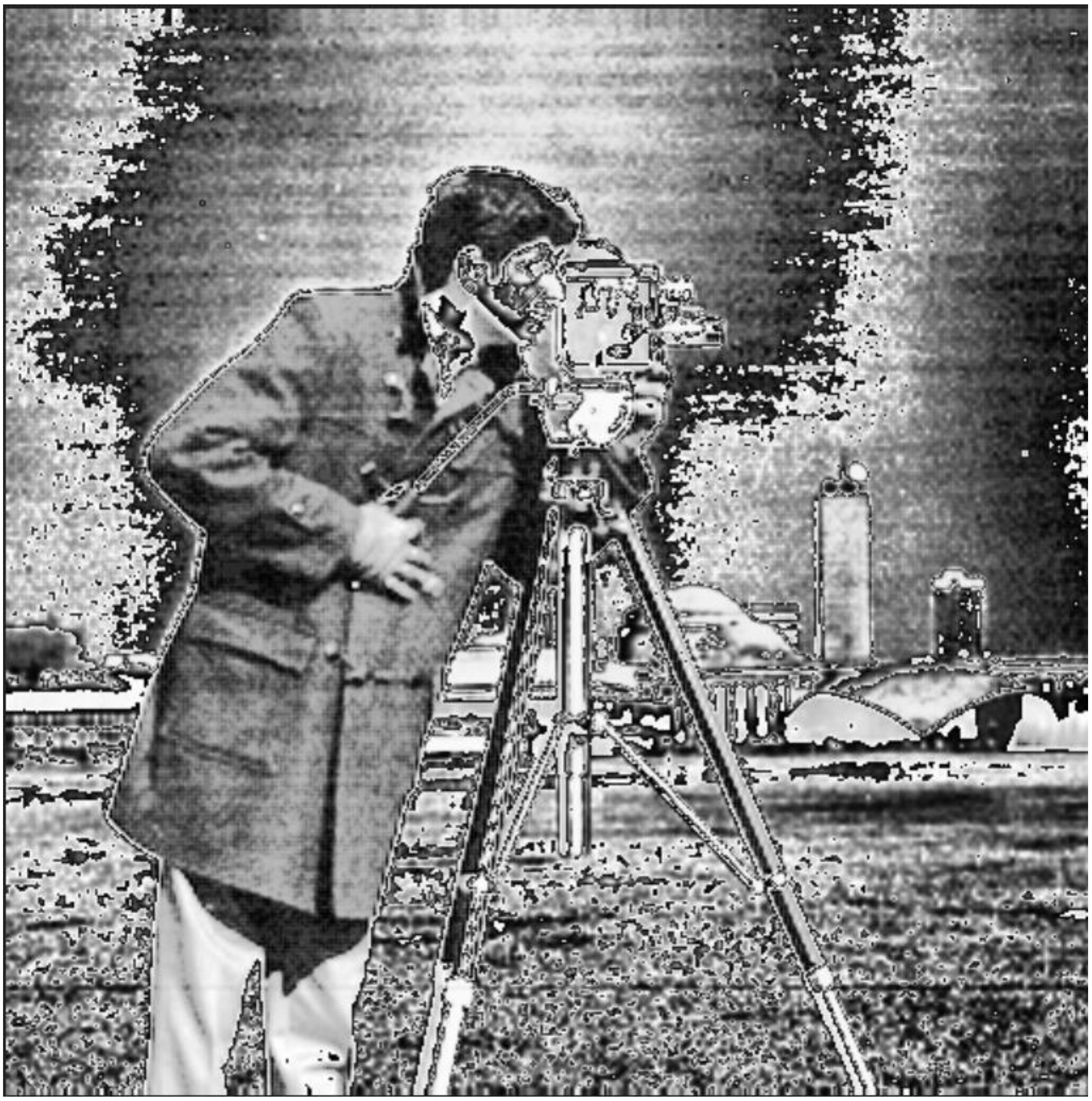
```
void showHistogram(Mat& image, string fileName) {

    int bins = 256;     // number of bins
    Mat histogram;      // for storing the histogram
    Mat canvas;                 // for displaying the histogram
    int hmax = 0;       // peak value for each histogram

    histogram = Mat::zeros(1, bins, CV_32SC1);

    for (int i = 0; i < image.rows; i++)
        for (int j = 0; j < image.cols; j++) {
            uchar val = image.at<uchar>(i, j);
            histogram.at<int>(val) += 1;
        }

    for (int j = 0; j < bins - 1; j++)
        hmax = histogram.at<int>(j) > hmax ? histogram.at<int>(j) : hmax;

    canvas = Mat::ones(125, bins, CV_8UC3);

    for (int j = 0, rows = canvas.rows; j < bins - 1; j++)
        line(canvas, Point(j, rows), Point(j, rows - (histogram.at<int>(j) * rows /
hmax)), Scalar(255, 255, 255), 1, 8, 0);

    imshow(fileName, canvas);

    }
```

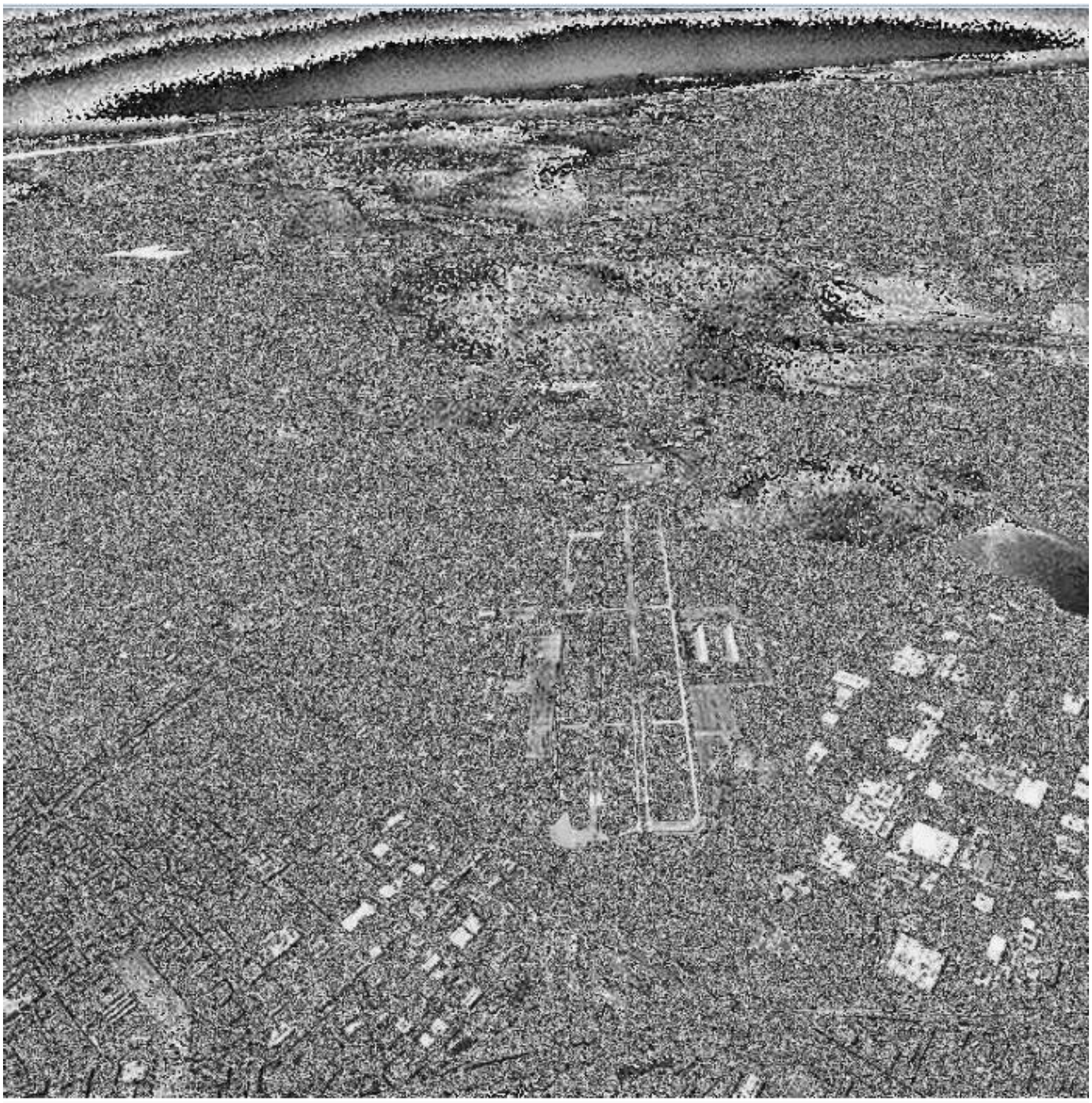
5) `histmatching:` This function is used to match the histogram of the source image and the target image. This function takes the cumulative probability distribution of source and target image. The function creates an array to store the position where the difference between cumulative function of source and target image is minimum.

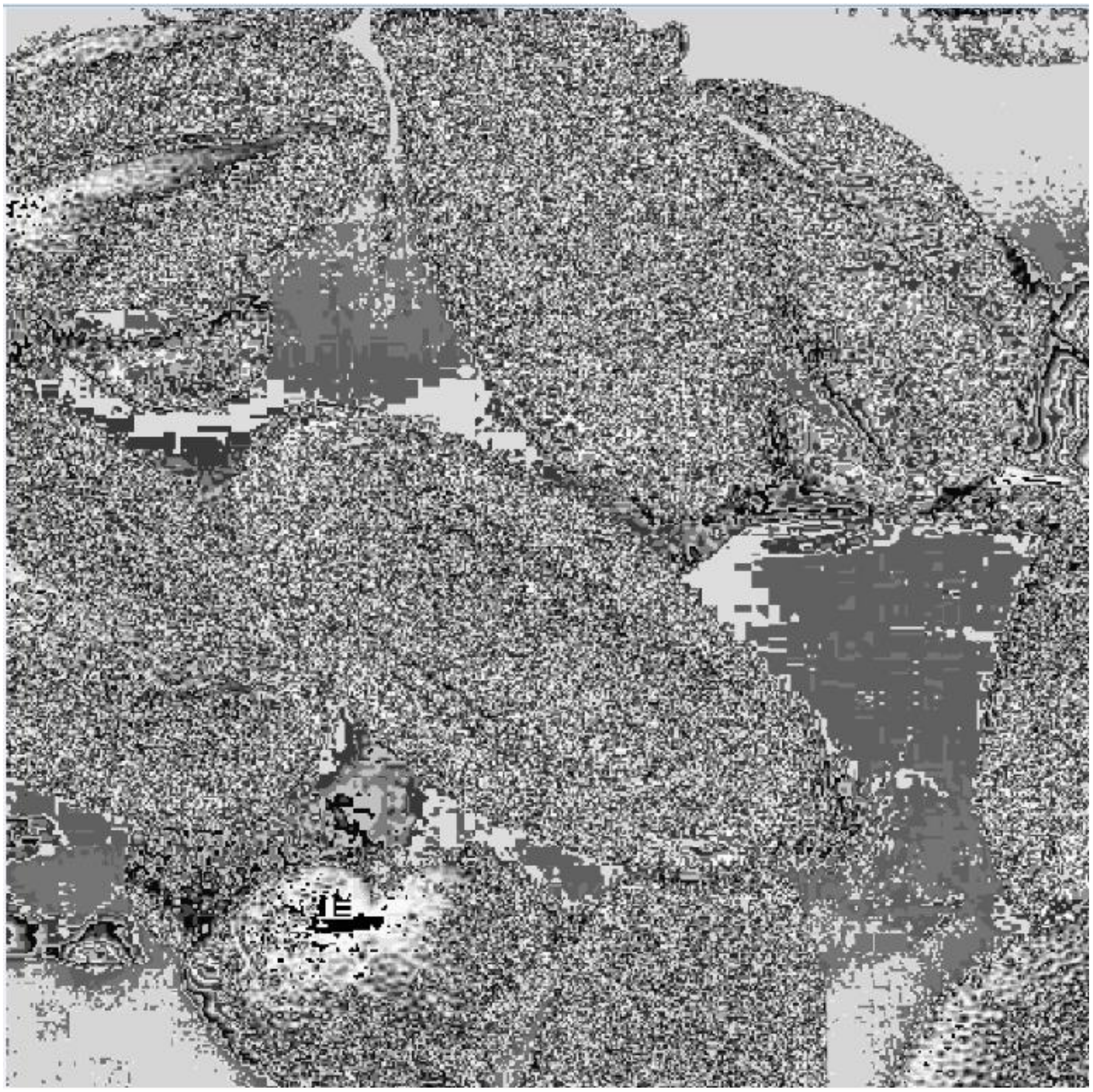
```
vector<double>histmatching(vector<double>source_cdf, vector<double>target_cdf)
{
    vector<double>histmat;
    vector<int>histmat1;
    histmat.resize(256, 0);
    histmat1.resize(256, 0);
    for (int i = 0; i < 256; i++)
    {
        double mini = (double)INT_MAX;
        for (int j = 0; j < 256; j++)
        {

            if ((abs(source_cdf[i] - target_cdf[j]) < mini))
            {
                //cout << source_cdf[i] << " " << target_cdf[j] << " " << source_cdf[i] -
target_cdf[j];
                mini = abs(source_cdf[i] - target_cdf[j]);
                // cout << mini << " ";
                histmat[i] = j;
            }
        }
    }
    //for (int i = 0; i < 256; i++) cout << histmat[i] << " ";
    return histmat;
   }
```

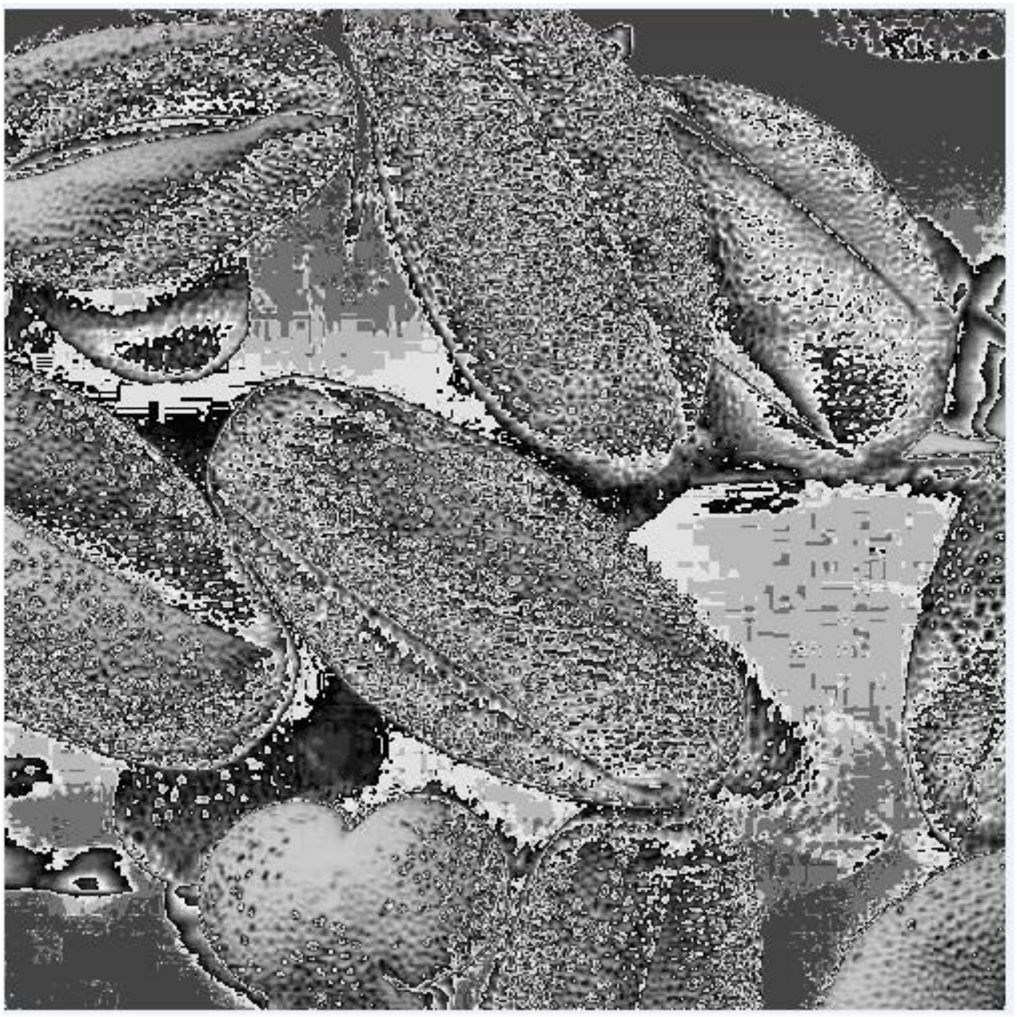
Output Image:

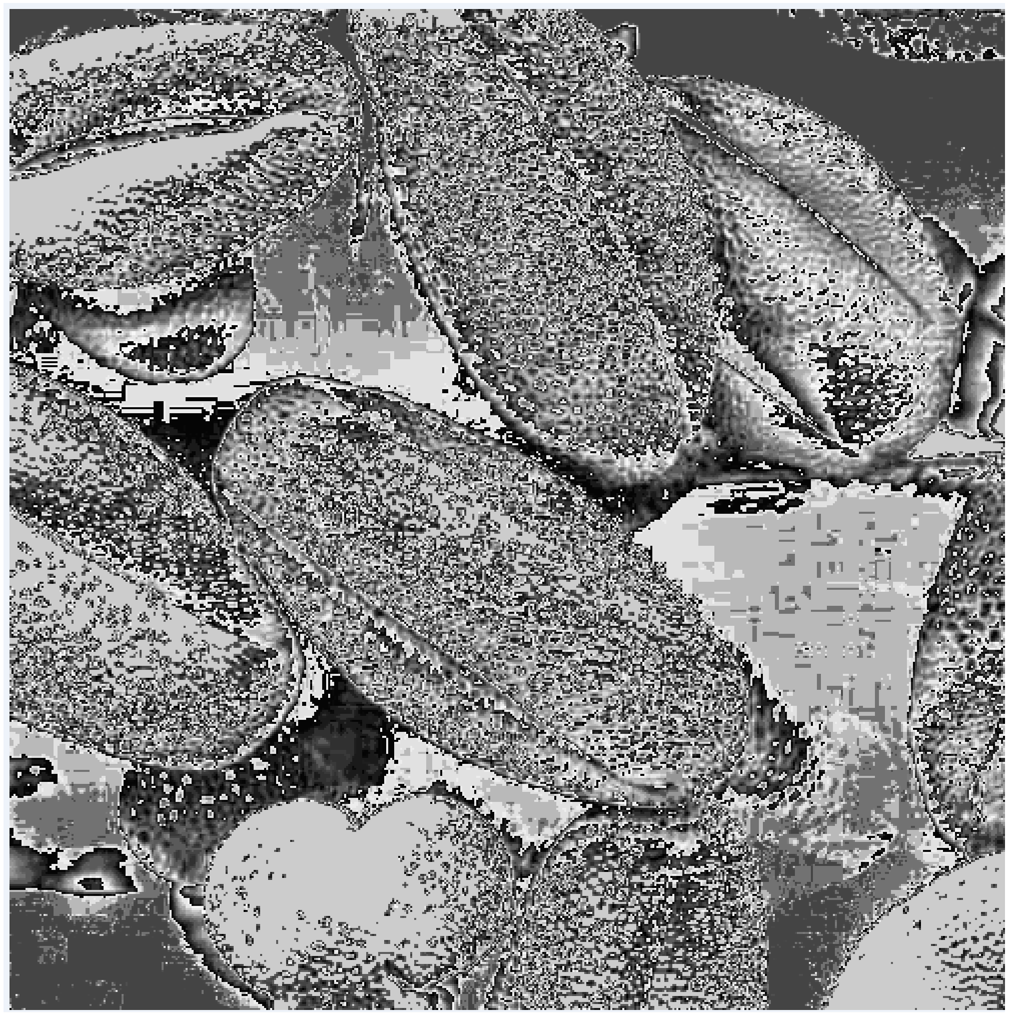













F-16
01568