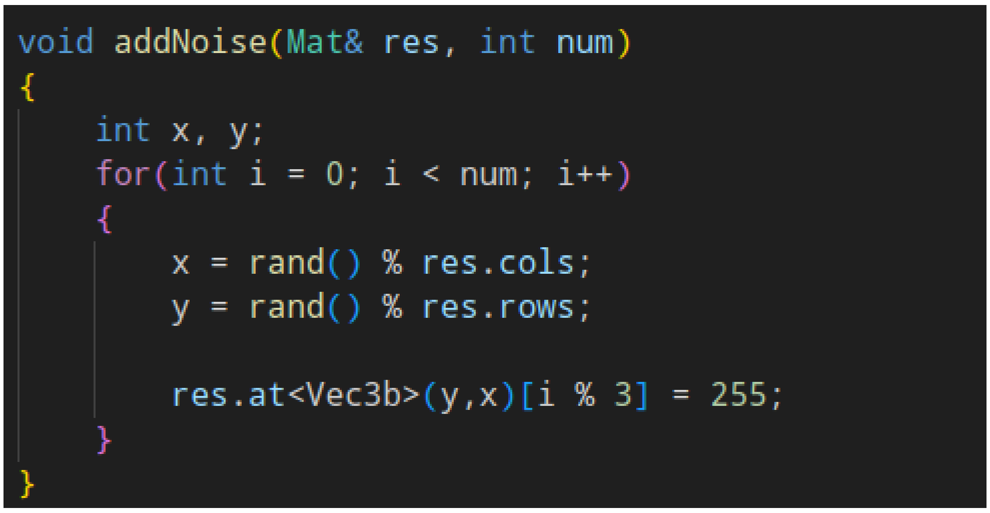
Digital Image Processing

Task 1:

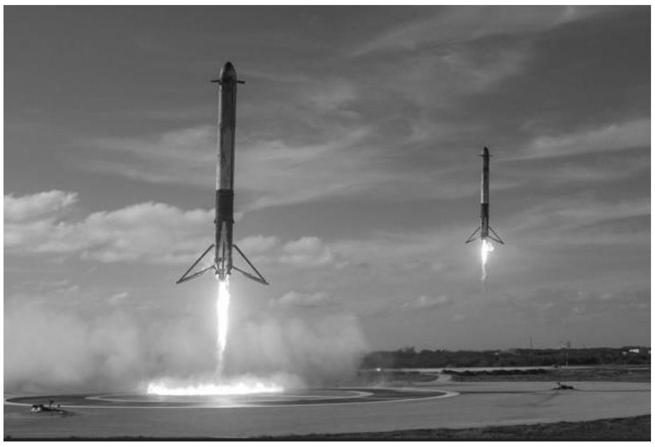
Using the random number generator and dividing it to number of columns and rows we can get random coordinate (in range of image dimensions) of pixel located at x, y. Chanel changes after each random generated location. For example: First random pixel would be blue, second random pixel would be green and third one would be red (BGR) and then the whole sequence starts again with the same pattern.



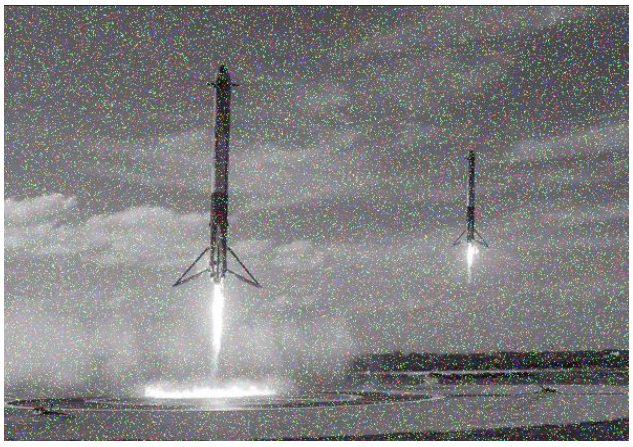
```
void addNoise(Mat& res, int num)
{
    int x, y;
    for(int i = 0; i < num; i++)
    {
        x = rand() % res.cols;
        y = rand() % res.rows;

        res.at<Vec3b>(y,x)[i % 3] = 255;
    }
}
```

The results of the program is provided below (Input number is 50000):

Original Image



Noisy Image

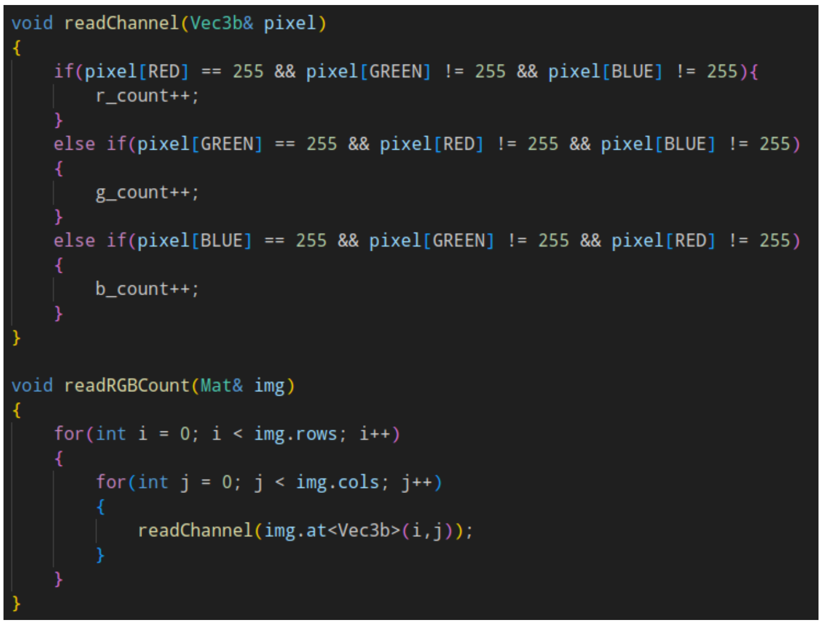


The version of image with noise added would be saved in file img1-res.png.

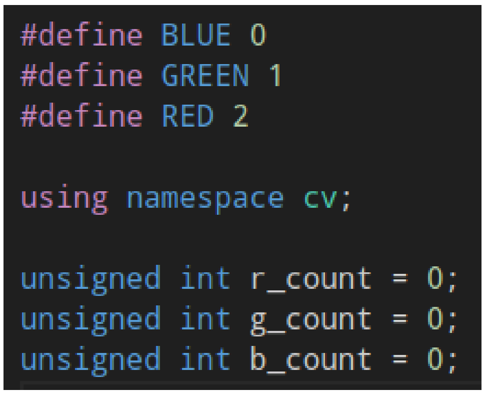
Important note: PNG file format is used to avoid loss of data.

Task 2:

The function loops through the individual pixels and checks whether one of channels has the maximum value of intensity(255) if channel with such intensity exists corresponding global variable would be incremented , if none of the channels has the highest intensity then the pixel is going to be ignored.

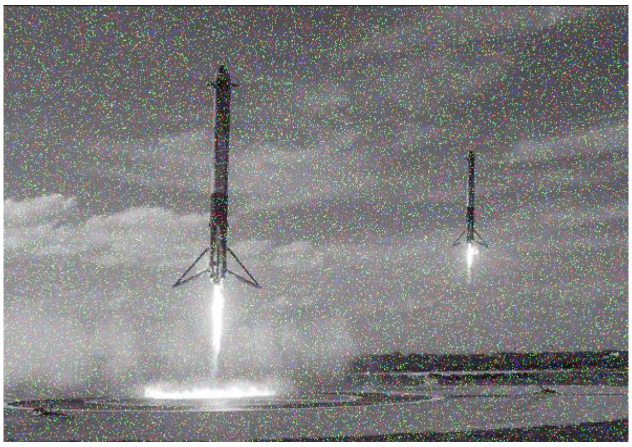


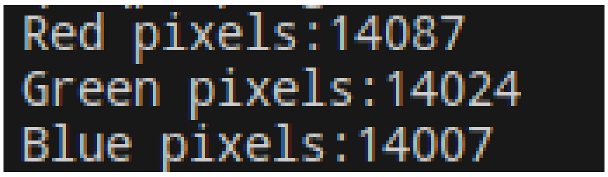
```
void readChannel(Vec3b& pixel)
{
    if(pixel[RED] == 255 && pixel[GREEN] != 255 && pixel[BLUE] != 255){
        r_count++;
    }
    else if(pixel[GREEN] == 255 && pixel[RED] != 255 && pixel[BLUE] != 255)
    {
        g_count++;
    }
    else if(pixel[BLUE] == 255 && pixel[GREEN] != 255 && pixel[RED] != 255)
    {
        b_count++;
    }
}

void readRGBCount(Mat& img)
{
    for(int i = 0; i < img.rows; i++)
    {
        for(int j = 0; j < img.cols; j++)
        {
            readChannel(img.at<Vec3b>(i,j));
        }
    }
}
```



```
#define BLUE 0
#define GREEN 1
#define RED 2

using namespace cv;

unsigned int r_count = 0;
unsigned int g_count = 0;
unsigned int b_count = 0;
```

As an input image, noisy image from previous task was used: (img1-res.png)





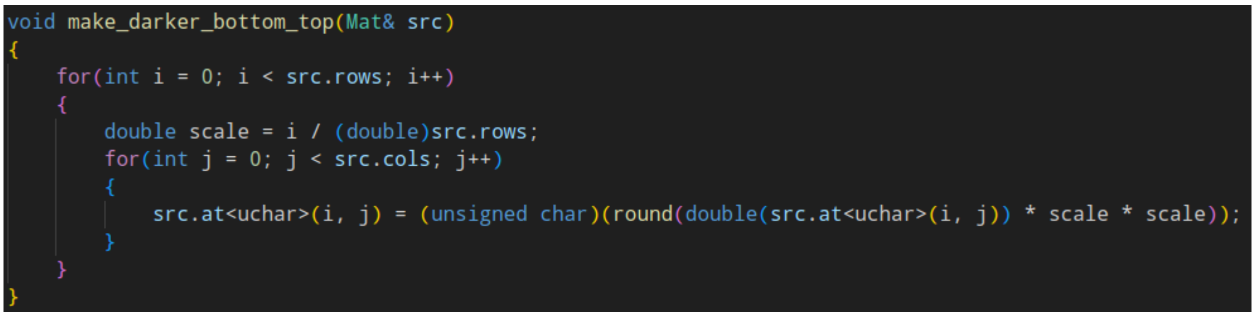
```
Red pixels:14087
Green pixels:14024
Blue pixels:14007
```

Due to the tendency of random number generator to repeat its previous values some pixels at given coordinates have two or all three channels changed to maximum intensity. For this reason, even though the input value was 50.000, total sum of red, green, and blue pixels doesn’t equal to 50.000.

Task 3:

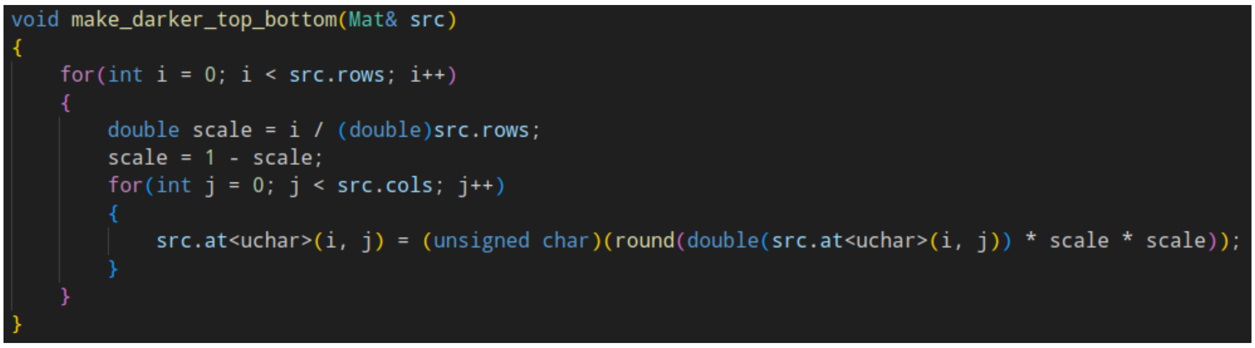
The first function loops through the gray-scale pixels of the image and multiples the value of pixel to the ratio of current column pixel located, to image height. As loops goes down the y axis increasing the ratio the intensity of the pixel would change less.

The function can be seen below (ratio is scale):



```
void make_darker_bottom_top(Mat& src)
{
    for(int i = 0; i < src.rows; i++)
    {
        double scale = i / (double)src.rows;
        for(int j = 0; j < src.cols; j++)
        {
            src.at<uchar>(i, j) = (unsigned char)(round(double(src.at<uchar>(i, j)) * scale * scale));
        }
    }
}
```

The second function does exactly the same except for after calculating ratio it would be subtracted by one making upper part of the image darker in comparison with lower.

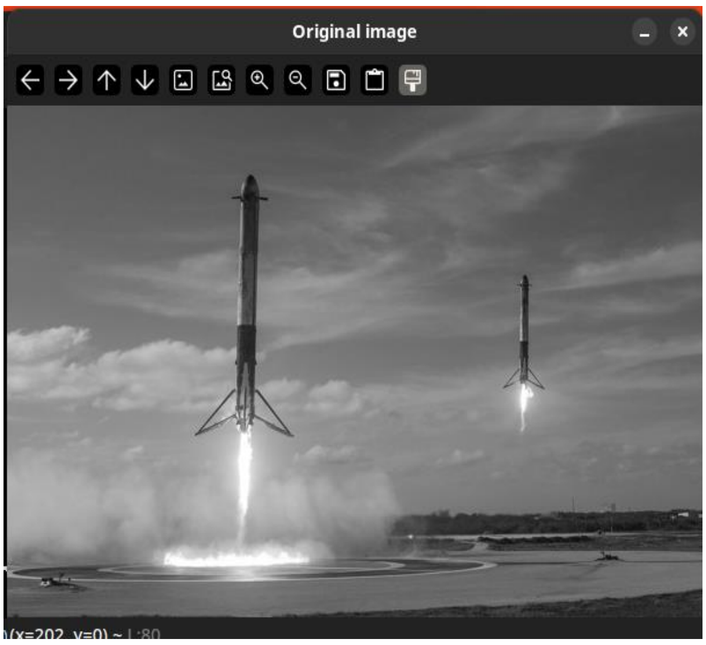


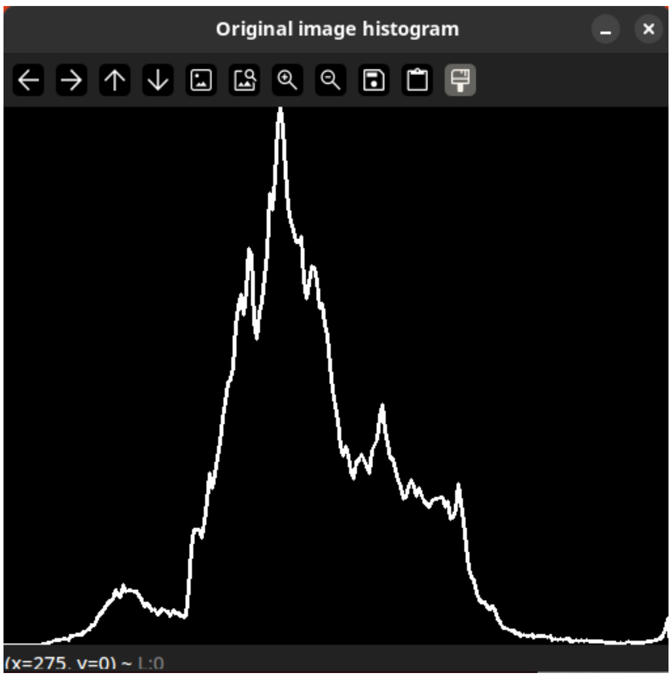
```
void make_darker_top_bottom(Mat& src)
{
    for(int i = 0; i < src.rows; i++)
    {
        double scale = i / (double)src.rows;
        scale = 1 - scale;
        for(int j = 0; j < src.cols; j++)
        {
            src.at<uchar>(i, j) = (unsigned char)(round(double(src.at<uchar>(i, j)) * scale * scale));
        }
    }
}
```

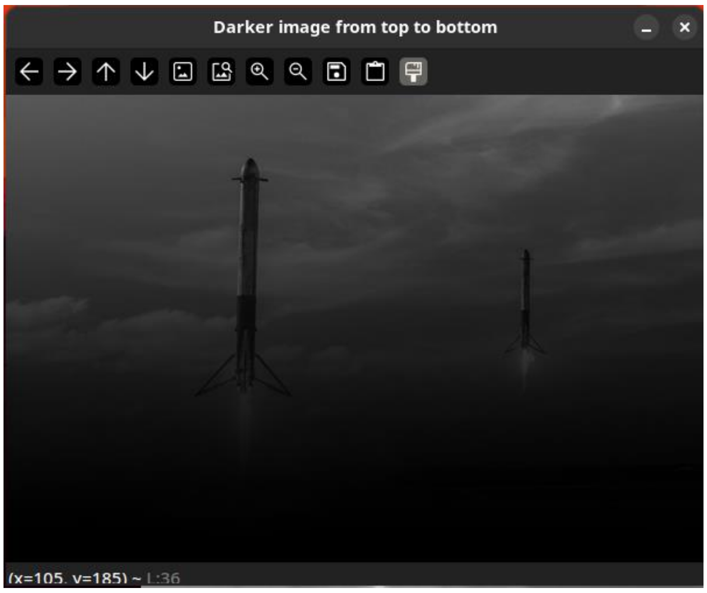
Note:

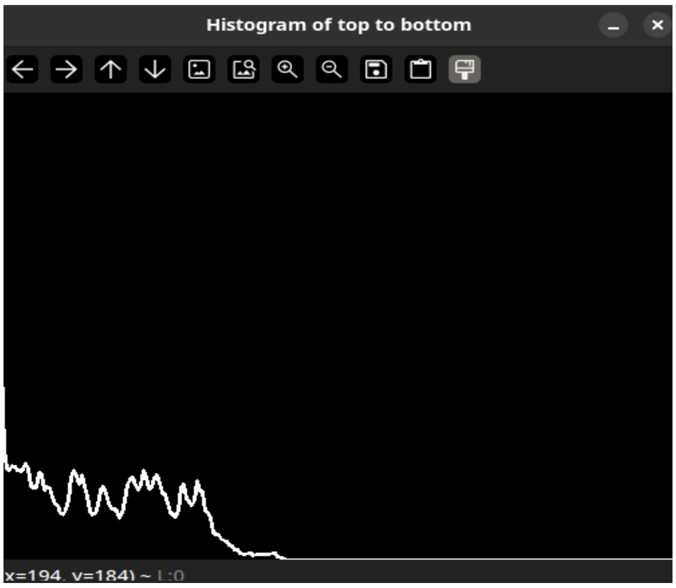
value of the pixel was multiplied to the ratio(scale) 2 times in order to make changes more noticeable!

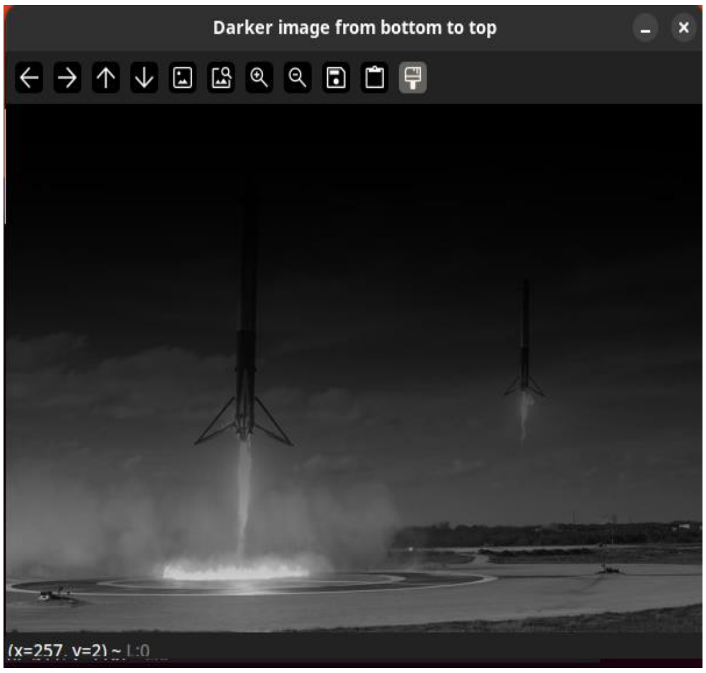
Histograms and Images themselves are shown below:

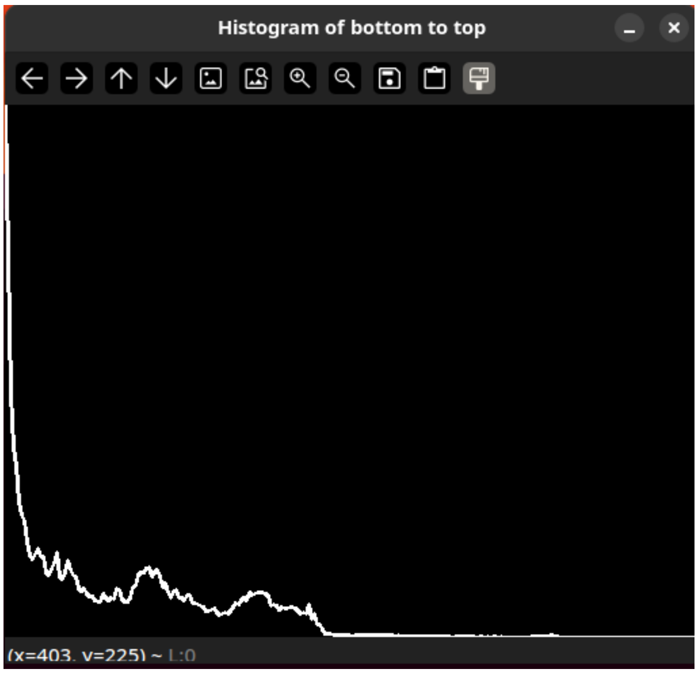

Original image
(x=202, y=0) ~ L:80


Original image histogram
(x=275, y=0) ~ L:0

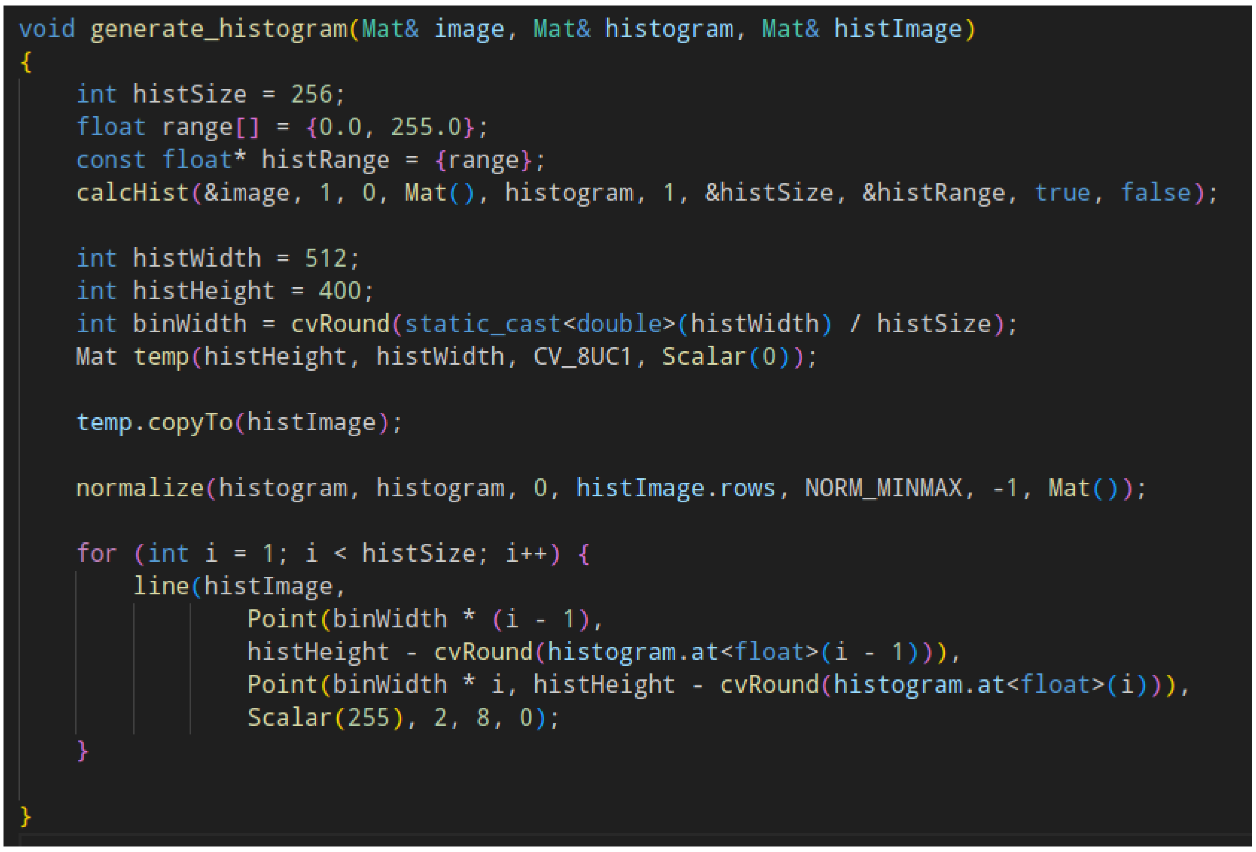

Darker image from top to bottom
(x=105, y=185) ~ L:36


Histogram of top to bottom
x=194, y=184) ~ L:0


Darker image from bottom to top
(x=257, y=2) ~ L:0

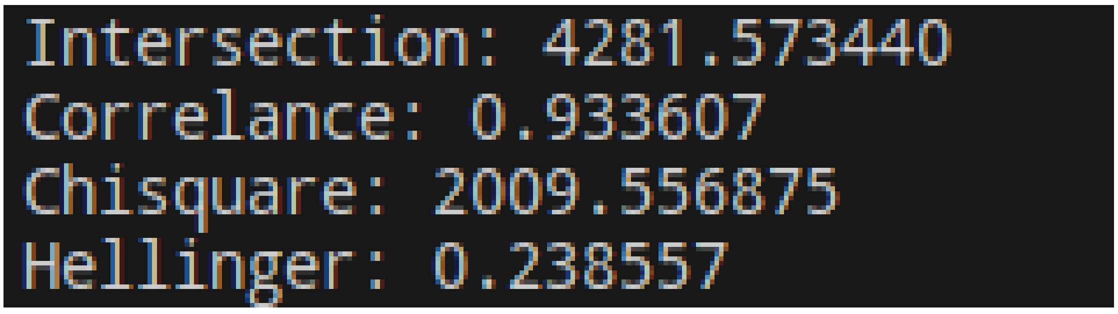

Histogram of bottom to top
(x=403, y=225) ~ L:0

Code for histogram generating:



```cpp
void generate_histogram(Mat& image, Mat& histogram, Mat& histImage)
{
    int histSize = 256;
    float range[] = {0.0, 255.0};
    const float* histRange = {range};
    calcHist(&image, 1, 0, Mat(), histogram, 1, &histSize, &histRange, true, false);

    int histWidth = 512;
    int histHeight = 400;
    int binWidth = cvRound(static_cast<double>(histWidth) / histSize);
    Mat temp(histHeight, histWidth, CV_8UC1, Scalar(0));

    temp.copyTo(histImage);

    normalize(histogram, histogram, 0, histImage.rows, NORM_MINMAX, -1, Mat());

    for (int i = 1; i < histSize; i++) {
        line(histImage,
                Point(binWidth * (i - 1),
                histHeight - cvRound(histogram.at<float>(i - 1))),
                Point(binWidth * i, histHeight - cvRound(histogram.at<float>(i))),
                Scalar(255), 2, 8, 0);
    }

}
```

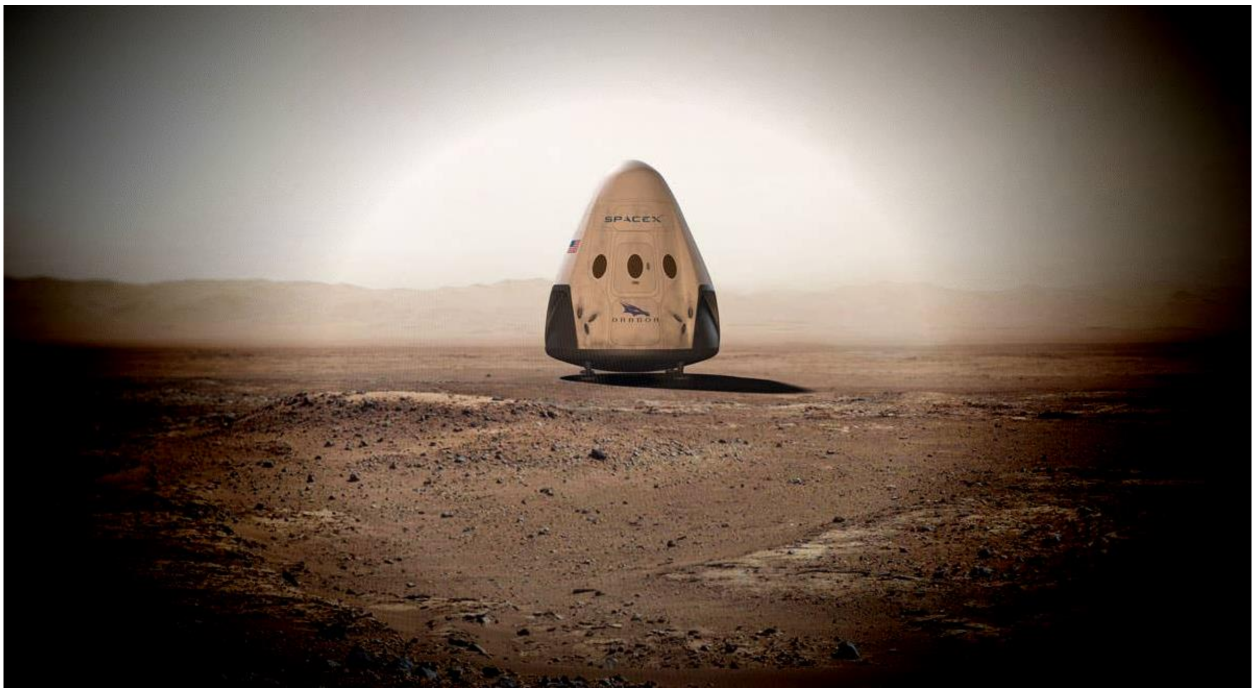
Result of histogram comparison between 2 dark images is given below:



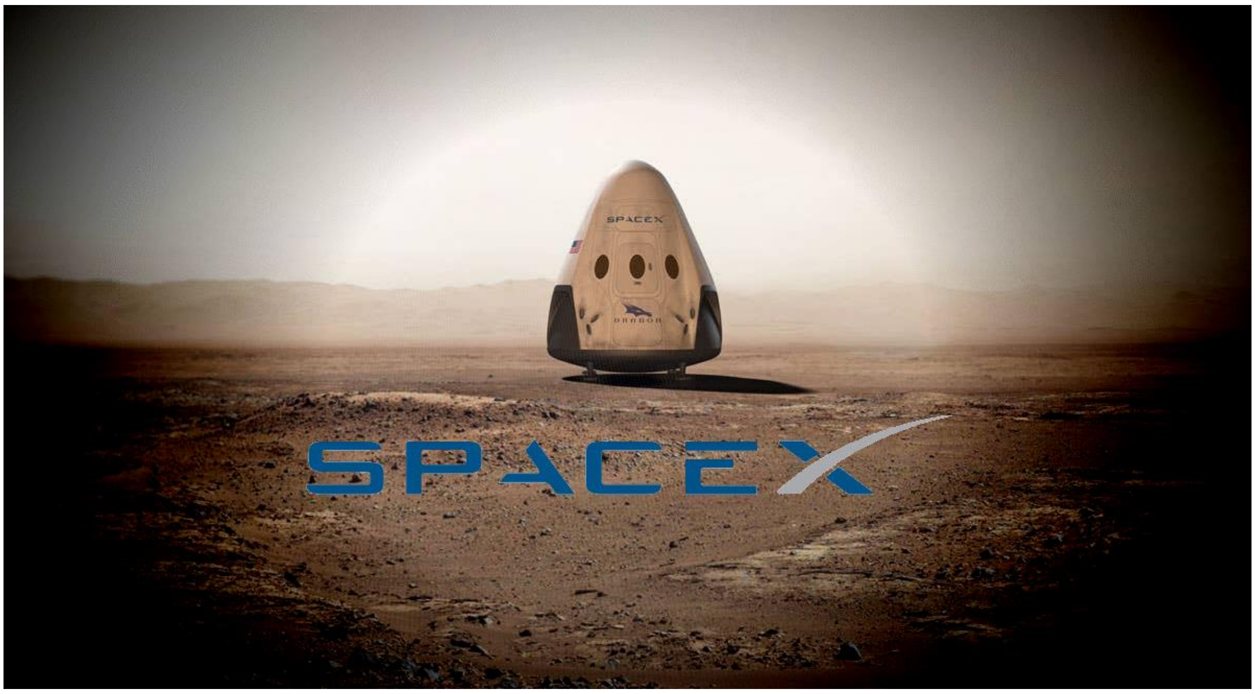
```
Intersection: 4281.573440
Correlance: 0.933607
Chisquare: 2009.556875
Hellinger: 0.238557
```

Task 4:

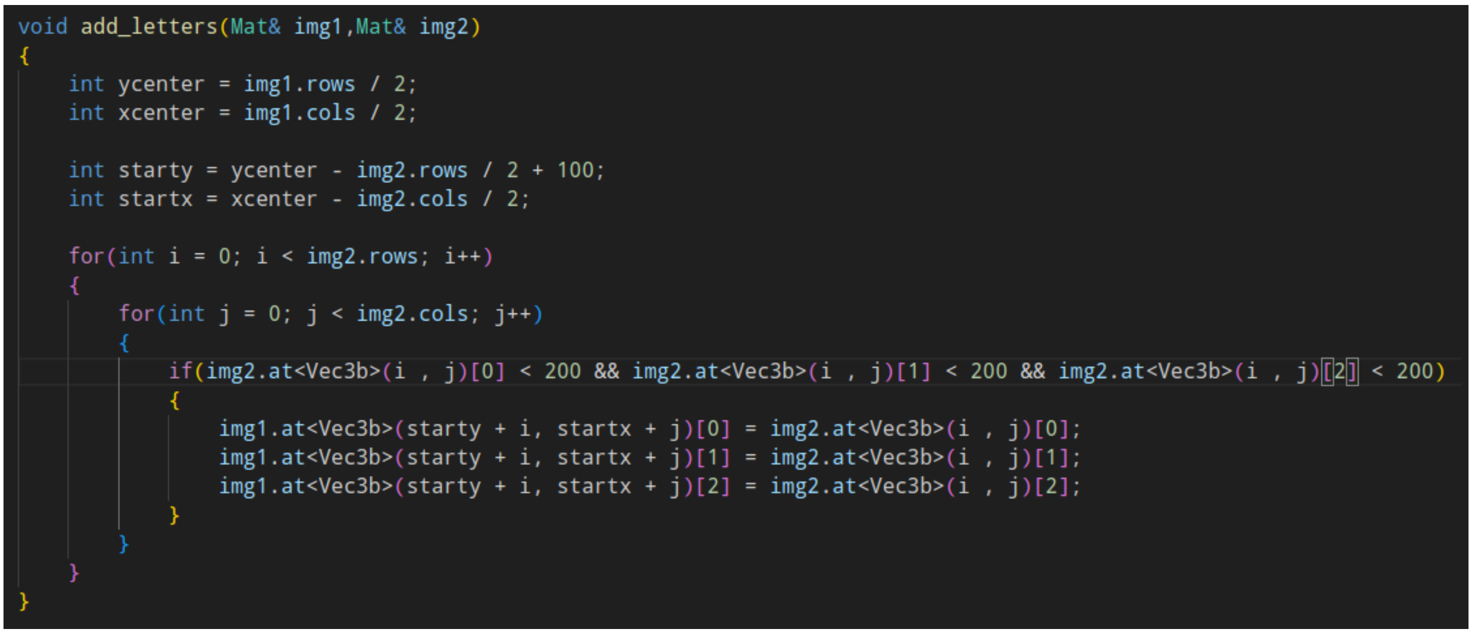
Firstly, the image3 and image4 were subtracted and saved onto dst1 matrix that gave the result shown below:



Then in dst1 matrix center point coordinates are calculated and subtracted from half dimensions of image5 in order to find starting point where pixels should be placed. After calculation the initial coordinates the pixels from image5 put onto dst1 matrix, all the bright pixels from image5 would be ignored, all remain pixels would replace the values on respective coordinate on matrix dst1. The final result will look like this:



The function adding text onto image is provided below:



```
void add_letters(Mat& img1,Mat& img2)
{
    int ycenter = img1.rows / 2;
    int xcenter = img1.cols / 2;

    int starty = ycenter - img2.rows / 2 + 100;
    int startx = xcenter - img2.cols / 2;

    for(int i = 0; i < img2.rows; i++)
    {
        for(int j = 0; j < img2.cols; j++)
        {
            if(img2.at<Vec3b>(i , j)[0] < 200 && img2.at<Vec3b>(i , j)[1] < 200 && img2.at<Vec3b>(i , j)[2] < 200)
            {
                img1.at<Vec3b>(starty + i, startx + j)[0] = img2.at<Vec3b>(i , j)[0];
                img1.at<Vec3b>(starty + i, startx + j)[1] = img2.at<Vec3b>(i , j)[1];
                img1.at<Vec3b>(starty + i, startx + j)[2] = img2.at<Vec3b>(i , j)[2];
            }
        }
    }
}
```

Student Name: Javid Guliyev

Student ID: 12235421