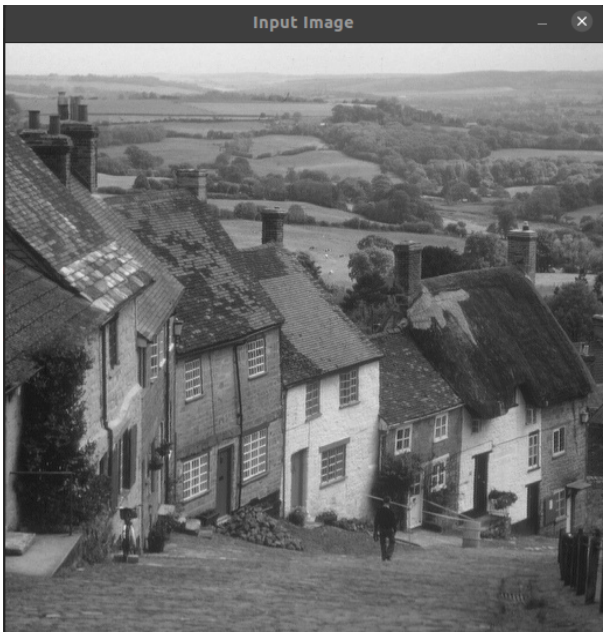
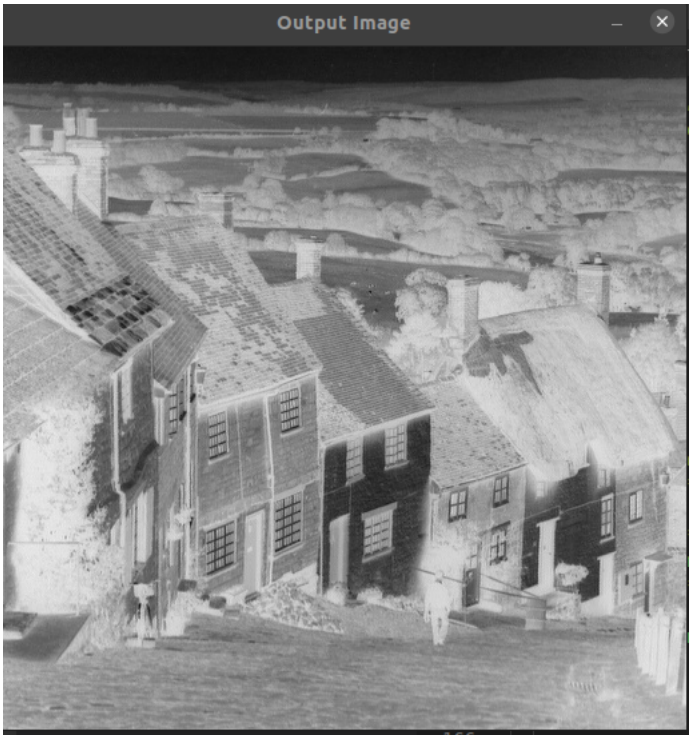
Alex Gaskins

6.1

Original (goldhill)



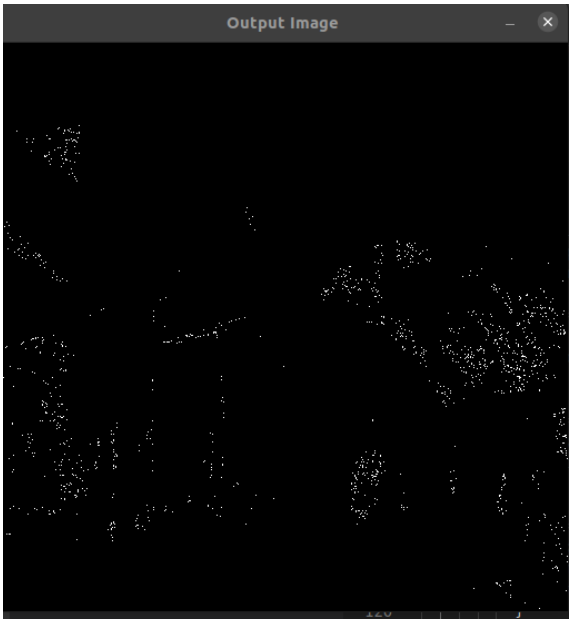
Black and White



Using Processing Function

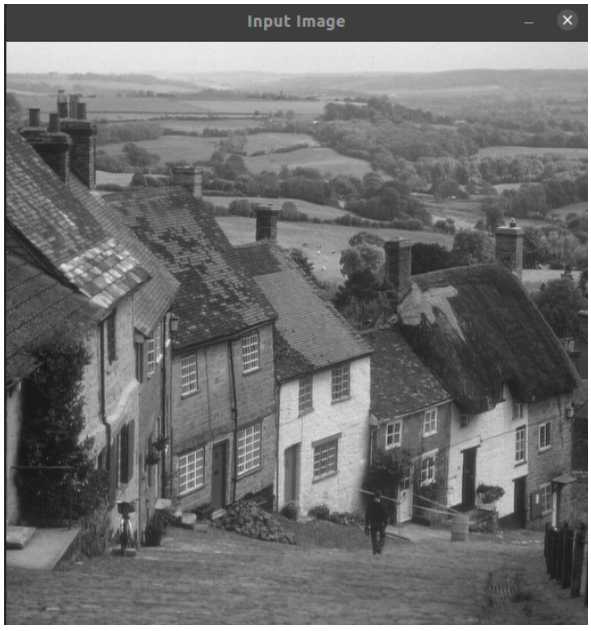
6.2

Original

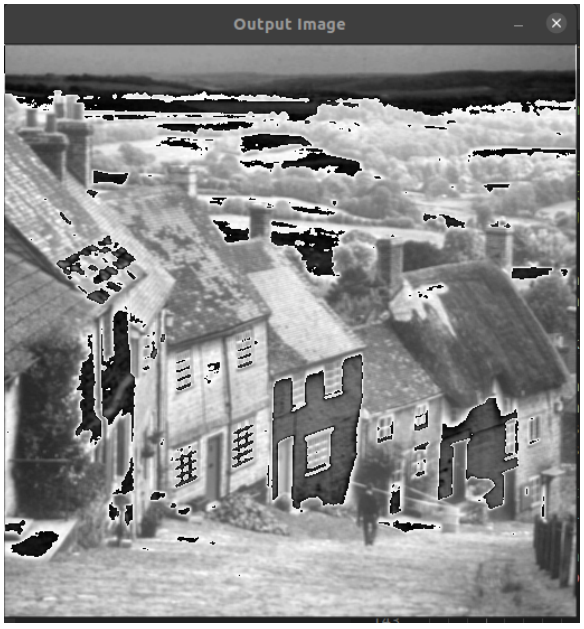


Histogram Function

6.3



Original



With 3x3 Filter

Code

```
int main(int argc, char* argv[])
{
    FILE* in, * out;
    int   j, k, width, height;
    int** image_in, ** image_out;
    float sum1, sum2;
    float new_T, old_T, delta_T;
    long count1, count2;


                                /* Some OpenCV here */
                                /*********************/

   Mat M_in = imread(argv[1]);
    Mat_<uchar> M_in_g(M_in.rows, M_in.cols);
    cvtColor(M_in, M_in_g, COLOR_BGR2GRAY);
    //cout << "height" << M_in_g.rows << endl;

    String windowName1 = "Input Image"; //Name of the window

    namedWindow(windowName1); // Create a window

    imshow(windowName1, M_in_g); // Show our image inside the created window.

    waitKey(0); // Wait for any keystroke in the window

    destroyWindow(windowName1); //destroy the created window

    height = M_in_g.rows;
    width = M_in_g.cols;

    /*************************************************************************/
```

```
    image_in = (int**)calloc(height, sizeof(int*));
    if (!image_in)
    {
        printf("Error: Can't allocate memmory!\n");
        return(1);
    }

    image_out = (int**)calloc(height, sizeof(int*));
    if (!image_out)
    {
        printf("Error: Can't allocate memmory!\n");
        return(1);
    }

    for (j = 0; j < height; j++)
    {
        image_in[j] = (int*)calloc(width, sizeof(int));
        if (!image_in[j])
        {
            printf("Error: Can't allocate memmory!\n");
        return(1);
        }

        image_out[j] = (int*)calloc(width, sizeof(int));
        if (!image_out[j])
        {
            printf("Error: Can't allocate memmory!\n");
            return(1);
        }

    }

    for (j = 0; j < height; j++)
        for (k = 0; k < width; k++)
            image_in[j][k] = M_in_g(j, k);
```

```
//Orginal Image Processing
    for (j = 0; j < height; j++)
        for (k = 0; k < width; k++)
        {
            image_out[j][k] = 255 - image_in[j][k];
    }
//histogram
        float hist[256];
        float eq_map[256];
        float one_pixel = 1.0 / ((float)width * height);

        for (j = 0; j < 256; j++) {
          hist[j] = 0.0;
          eq_map[j] = 0.0;
        }

        for (j = 0; j < height; j++)
          for (k = 0; k < width; k++) {
            hist[image_in[j][k]] += one_pixel;
          }

        for (j = 0; j < 256; j++)
          for (k = 0; k < j + 1; k++) {
            eq_map[j] += hist[k];
            eq_map[j] = floor(eq_map[j] * 255.0);
          }

        for (j = 0; j < height; j++)
          for (k = 0; k < width; k++) {
            image_out[j][k] = (int)eq_map[image_in[j][k]];
          }
```

```
//filter
        int kernalRows = 3;
        int kernalCols = 3;

        double kernal[kernalRows][kernalCols] = {{0.3, 0.3, .3}, {0.2, .2,0.2} ,{0.1, 0.1, 0.1}};

        int kCenterX = kernalCols / 2;
        int kCenterY = kernalRows / 2;
        int rows = width;
        int cols = height;

        for (int i = 0; i < rows; ++i) {
          for (int j = 0; j < cols;++j) {
            for (int m = 0; m < kernalRows;++m) {
              int mm = kernalRows - 1 - m;
              for (int n = 0; n < kernalCols;++n) {
                int nn = kernalCols - 1 - n;

                int ii = (i + (kCenterY - mm));
                int jj = (j + (kCenterX - nn));
                if(ii >=0 && ii <rows && jj >=0 && jj < cols) {
                  image_out[i][j] += image_in[ii][jj] * kernal[mm][nn];
                }
              }
            }
          }
        }
        /*****************************************************************/
        /* Image Processing ends                                         */
        /*****************************************************************/
```

```
        /*****************************/
        /* Some OpenCV here */
        /*****************/

        Mat_<uchar> M_out(height, width);
    for (int ii = 0; ii < height; ii++)
        for (int jj = 0; jj < width; jj++)
            M_out(ii, jj) = image_out[ii][jj];


 String windowName2 = "Output Image"; //Name of the window

    namedWindow(windowName2); // Create a window

    imshow(windowName2, M_out); // Show our image inside the created window.

    waitKey(0); // Wait for any keystroke in the window

    destroyWindow(windowName2); //destroy the created window


        bool isSuccess = imwrite(argv[2], M_out); //write the image to a file as JPEG
    //bool isSuccess = imwrite("MyOutputImage.png", M_out); //write the image to a file as PNG
    if (isSuccess == false)
    {
        cout << "Failed to save the image" << endl;
        //cin.get(); //wait for a key press
    return 1;
    }



        for (j = 0; j < height; j++)
    {
        free(image_in[j]);
        free(image_out[j]);
    }
    free(image_in);
    free(image_out);
}
```