754 PROJECT

Edge Detection using Skellam distribution as a sensor noise model

Hitesh Kumar Punna, Sudhansh Peddabomma

190050093,190050118

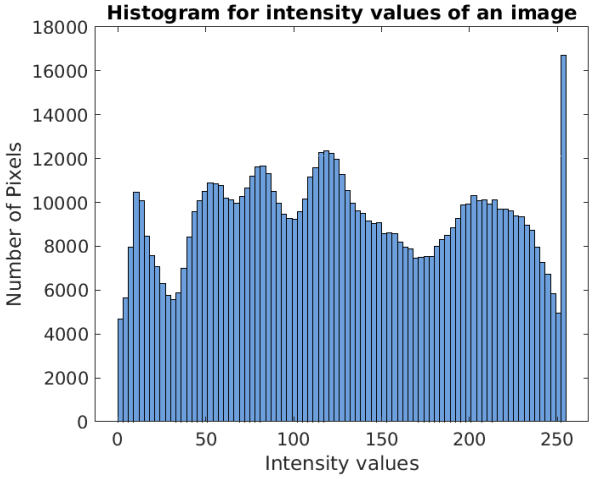
Instructions

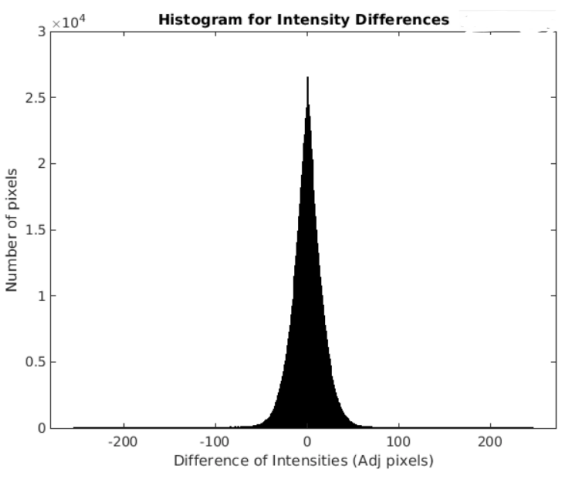
There are 2 folders in the compressed submission.

- Edge Detection
  1. *Skellam.m* is the main file which is needed to run to generate the results.
  2. *Test.m* and *Patches.m* are used for Designing/ Constructing the synthetic Image data set. The results are stored in the Tests and Patches folder respectively.
  3. *Trans.m* is a function file used for calculating confidence intervals.
  4. *Pmf.m* is a function file used for calculating pmf of the Skellam distribution.
  5. Output Graphs and Images are added in the *Results* folder.
- Background Subtraction
  This folder has a similar structure as above and it performs Background Subtraction. Run *Skellam.m* to obtain the results for a synthetically generated image (generated by this file itself).

Intuition behind the Skellam distribution

We plot the histogram of the intensities of an image and the histogram of difference of intensities of an image. Notice that the former histogram is random but the latter seems to follow a specific distribution. As we shall see, this is the Skellam distribution.
Note. We see a sudden rise in the first histogram at high intensity. This is because the image has poisson noise over an image with random uniform patches. This noise saturates at 255 and thus causes the peak.





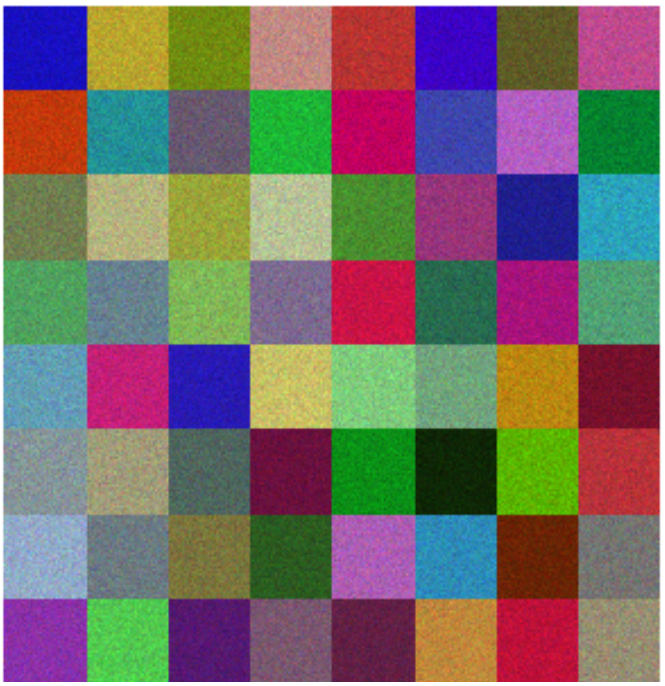
Dataset

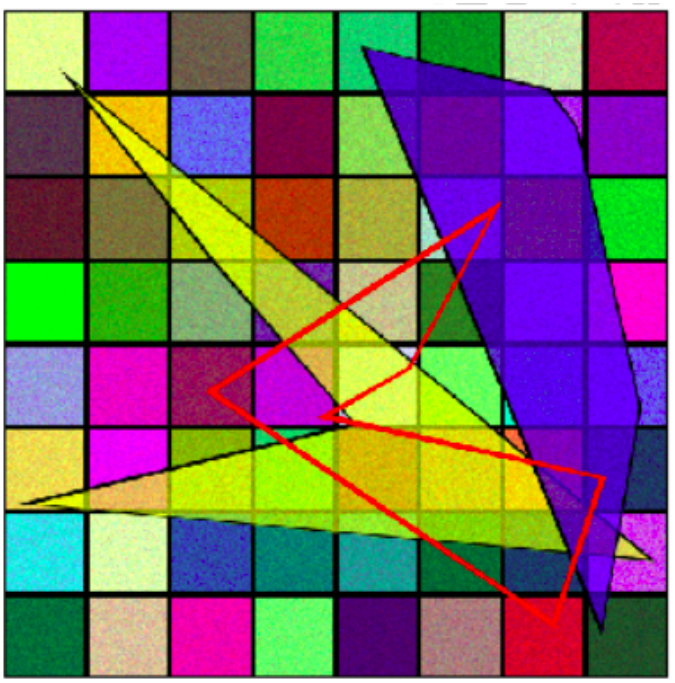
We compiled a synthetic dataset and generated training images by adding **Poisson noise** to an image with homogenous patches. We used “.tiff” and “.png” images as they utilise lossless compression algorithms. Other typical formats like “.jpeg” use DCT coefficients to compress the image and we lose the hypothesis of Poisson noise followed by intensities.

The following images are examples of the pictures generated by our code. We used the left image to calculate of μ_s and σ_s. These values are used to estimate the Skellam parameters.

The right image is used for edge detection via μ_s and σ_s values derived from the training data set.

Training images can be generated via *patch_generator.m* and the Testing images can be generated via *test.m*.
Note. As the gain parameter does not change because of addition of Poisson noise to homogenous uniform patches, both the images have the same Skellam parameters for a given intensity.





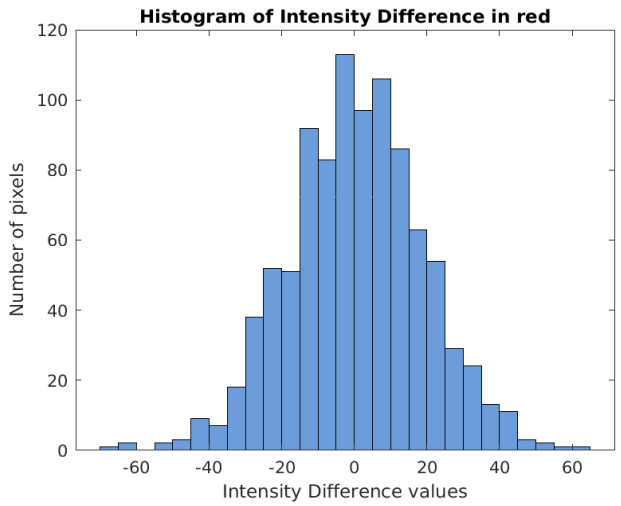
Use *patch_generator.m* to generate images on the left.
Use *test.m* to generate images on the right.
Skellam.m generates the images by itself, or you can manually add the images.

Validation of Skellam distribution

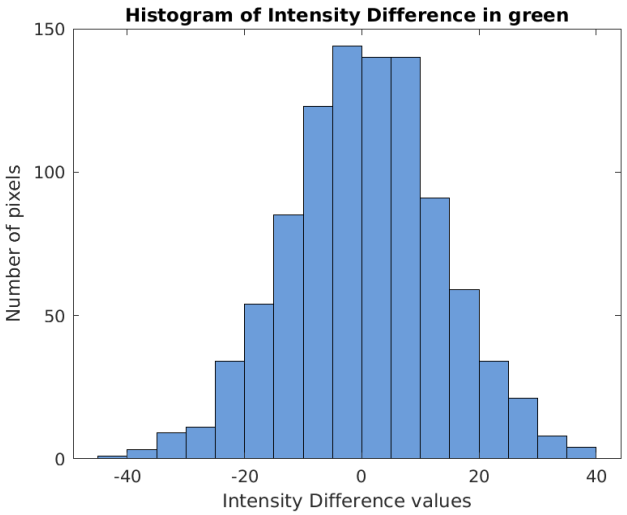
As done in the Research paper we found μ_s and σ_s by taking homogenous patches in the image.

The following are the distribution for *difference values* in a particular path for different colors. They approximately represent a Skellam distribution.

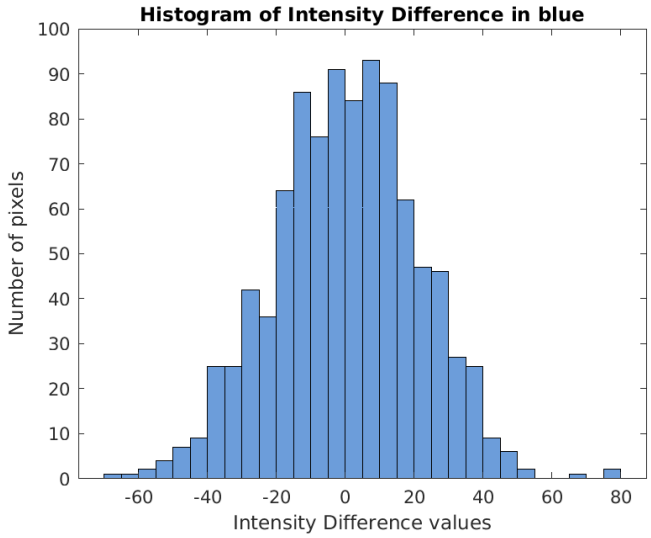
Color Red



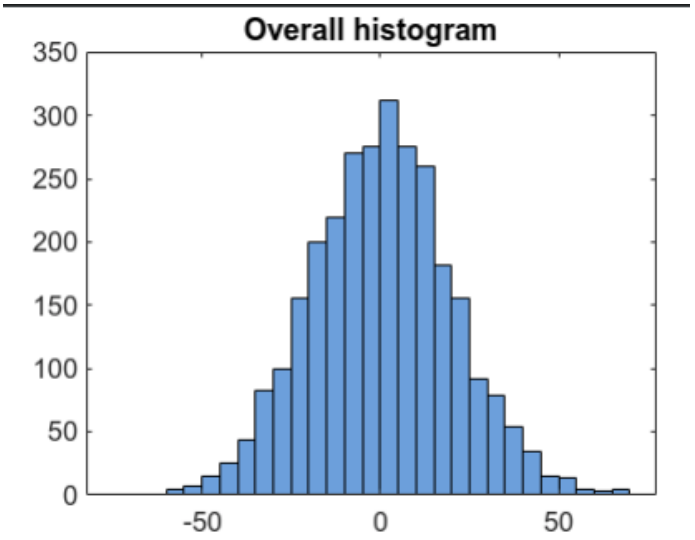
Color Green



Color blue

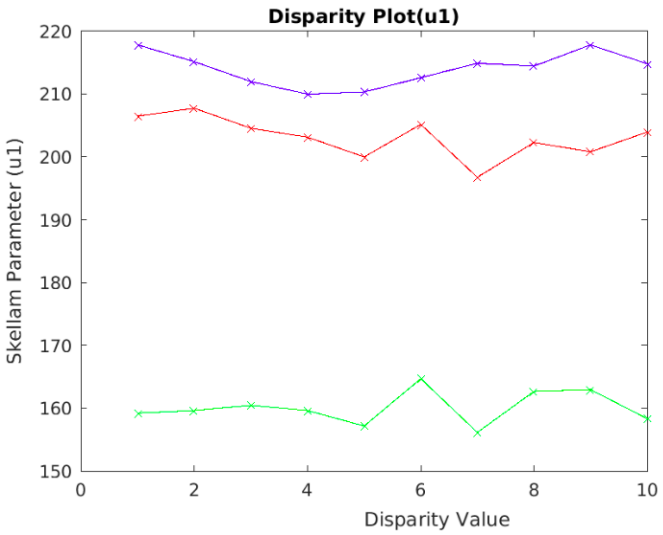


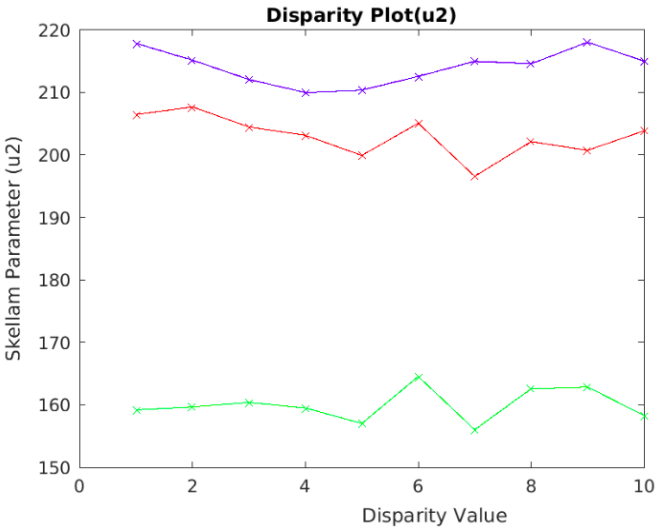
Combined Histogram



Disparity Plots

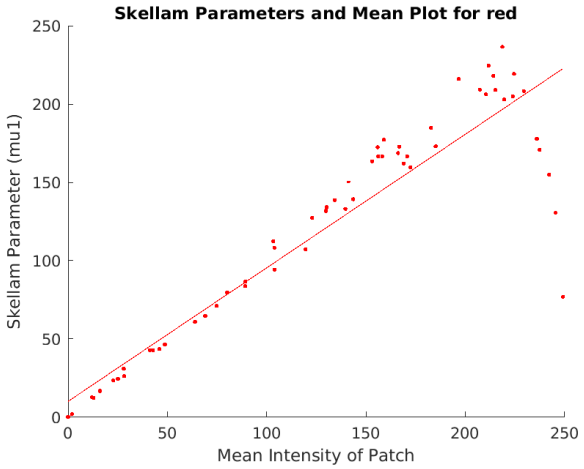
The Plots obtained by varying (dx, dy) to calculate the Skellam parameters. As we can see, the lines are almost horizontal. This means that the Skellam parameters are independent of dx and dy.

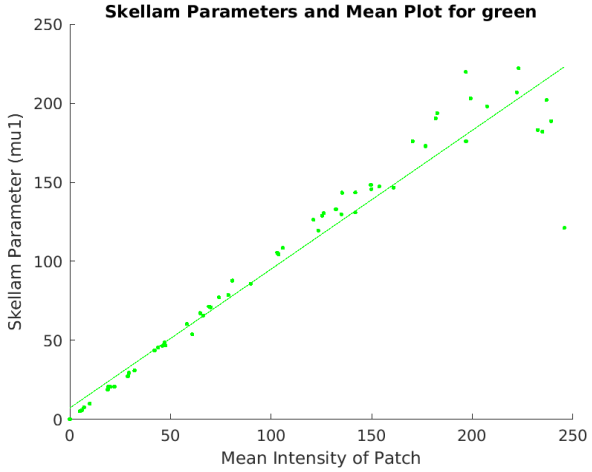


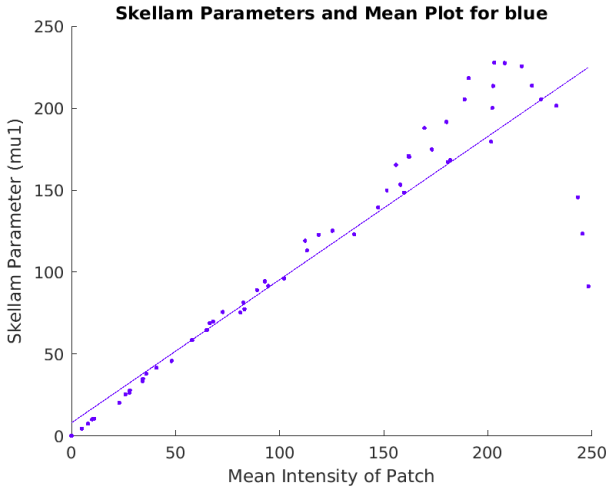


Skellam Parameters and Mean fit (Intensity - Skellam line)

We obtain the Skellam Parameters μ_1 and μ_2 via mean of the patch. Notice that there are outliers when the mean intensity is high, this is because of saturation of intensity values.



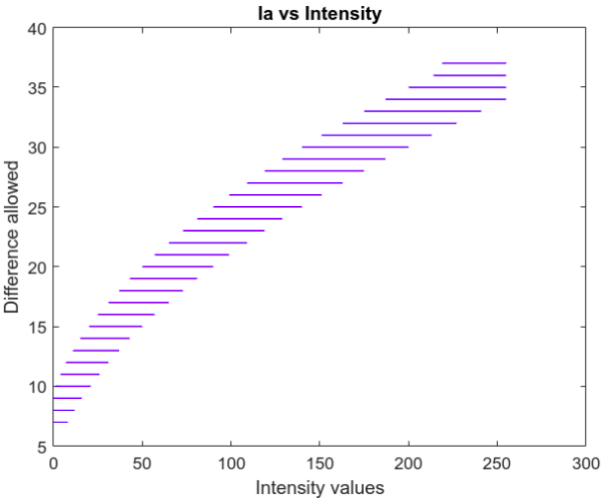




This data helps us to validate the idea of Skellam distribution for image noise.

Acceptance Region

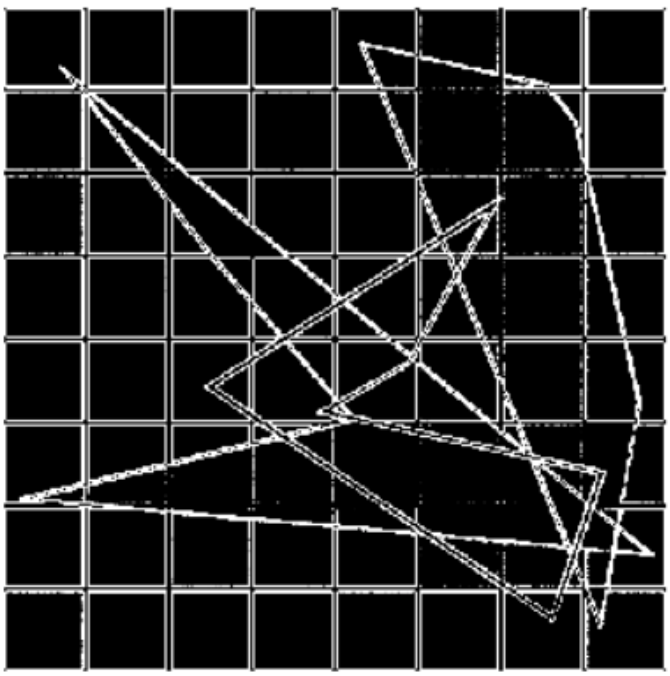
We find the confidence intervals defined by a parameter α for each intensity value in the image.



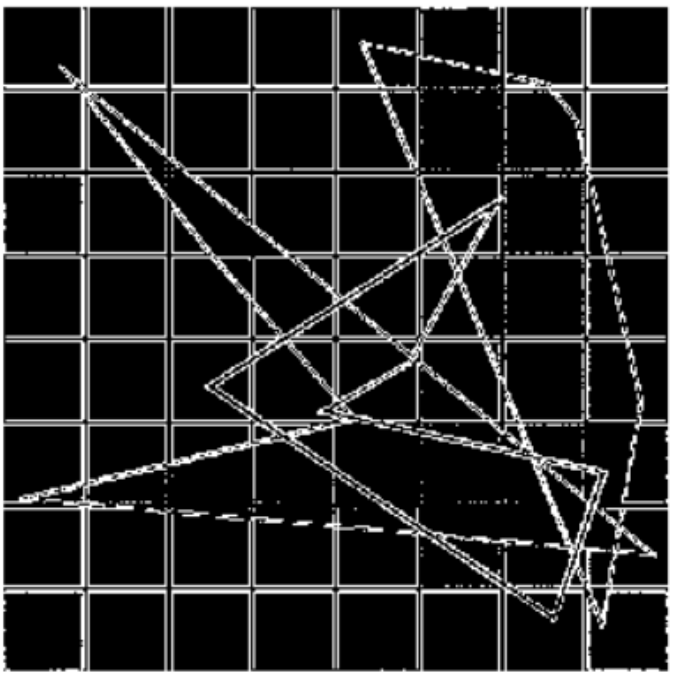
We show the confidence intervals for each intensity value in the above figure. Notice the clipping at intensity 255.

Edge Detection

The confidence intervals are used as a threshold to remove the non edged pixels. We find the confidence intervals via the PMF of the Skellam distribution as estimated by the intensity of the particular pixel. We also use non-maximum suppression to reduce false positives and makes the process robust.
The images after removing the non edged pixels look like :

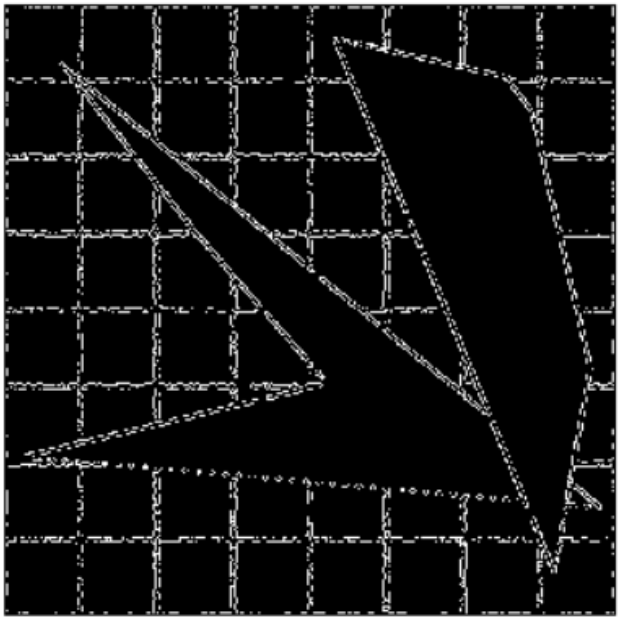


With Confidence Interval

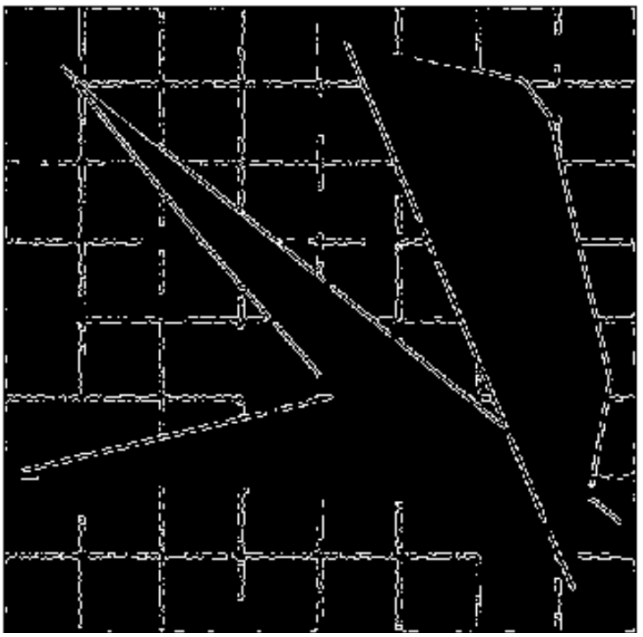


With Non Maximum Suppression

For comparison, this is how Canny edge detection looks for the above image. Canny Edge Detection has a rounding effect near the corners because of Gaussian Blurring. Also, the internal edges inside the filled polygons of the grid are not detected by this method. Edge Detection via Skellam noise modelling is able to outperform Canny algorithm as it is taking advantage of the noise distribution.



Low Threshold

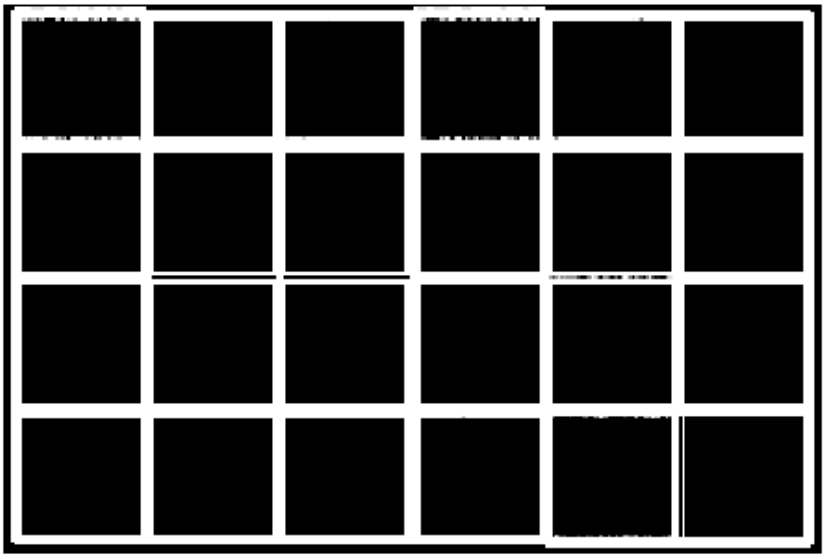


High Threshold

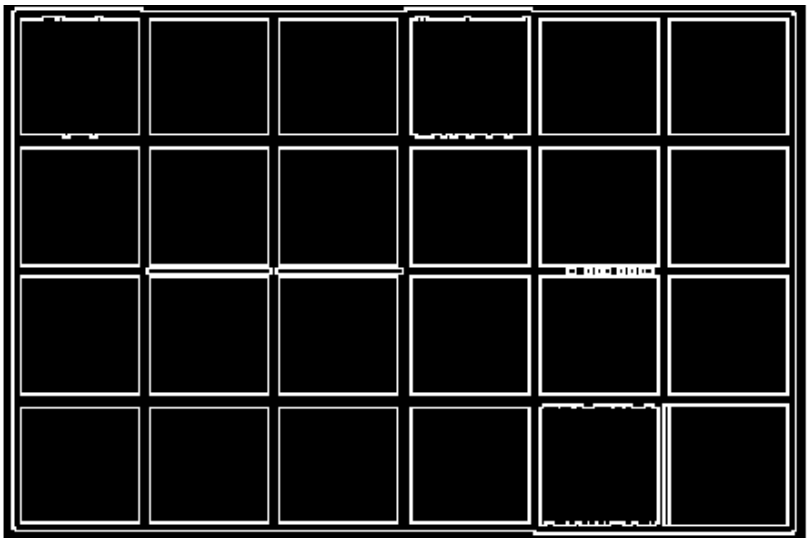
More Results

Original Image:-

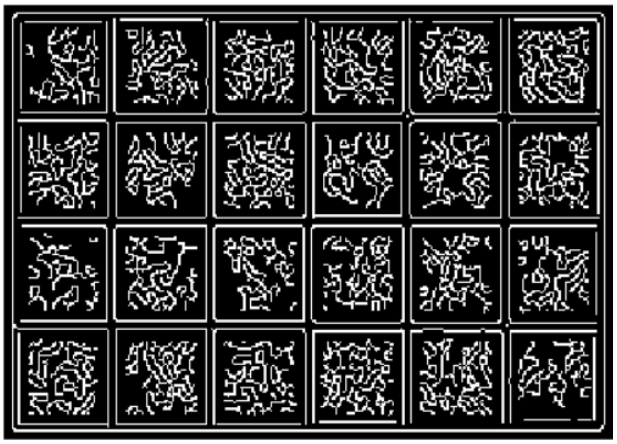
Edge Detection via Image generated using Confidence Intervals:-



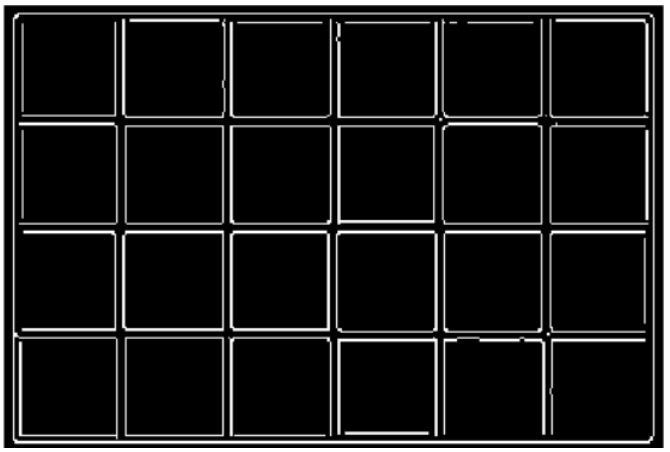
2.Images after Non-Maximum Suppression will look like the following



3.The canny results for above image on using low and high threshold will look like:-



Low Threshold



High Threshold

Observations

---

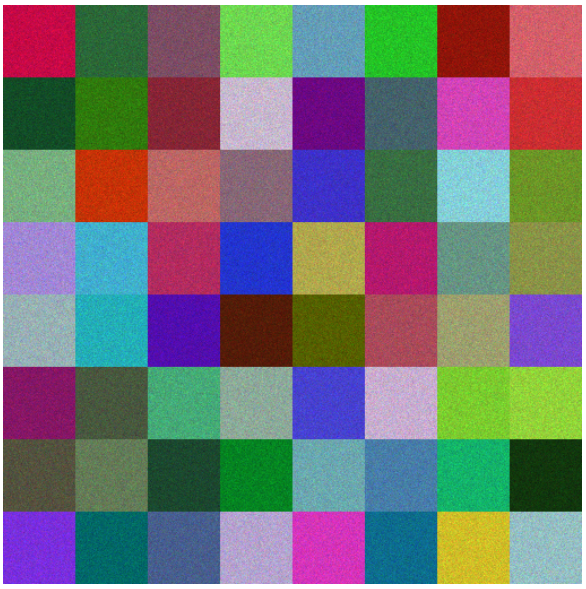
1. As we go towards high Intensity the poisson distribution saturates the intensity value (The pixel intensity cannot go beyond 255 for 8-bit images) and linear fit is not suitable in such scenarios.
2. Corners are detected with better accuracy. This is because Canny edge detection uses Gaussian Blurring and that distorts the corners.
3. Canny edge detection fails to give good results in many scenarios. The following are the drawbacks of the Canny edge detector, when compared with Skellam Noise modelling.
    a. Some edges are missing
    b. Intersecting edges are not getting shown
    c. Edge detection is very noisy

4. Canny edge detection is mainly used for grayscale images whereas our method works for color images.

As an extension to the above ideas, we apply this methodology for Background Subtraction as mentioned in the next section.

Background Subtraction

We can make us of the noise modelling and perform Background Subtraction in images. The intuition is quite similar and the results are given below.



Original Image

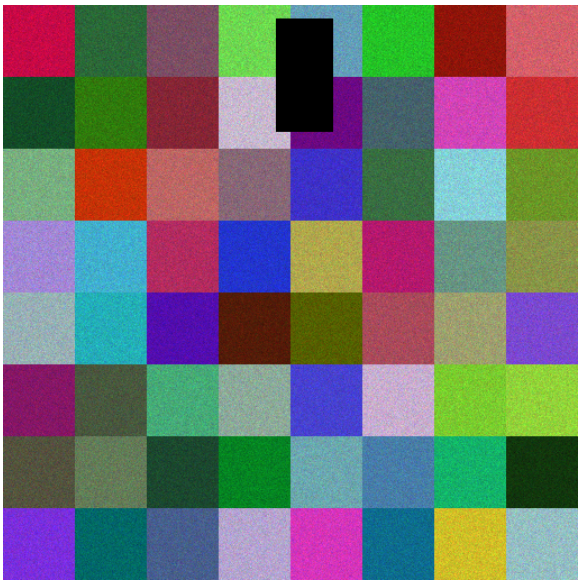


Image with an Object

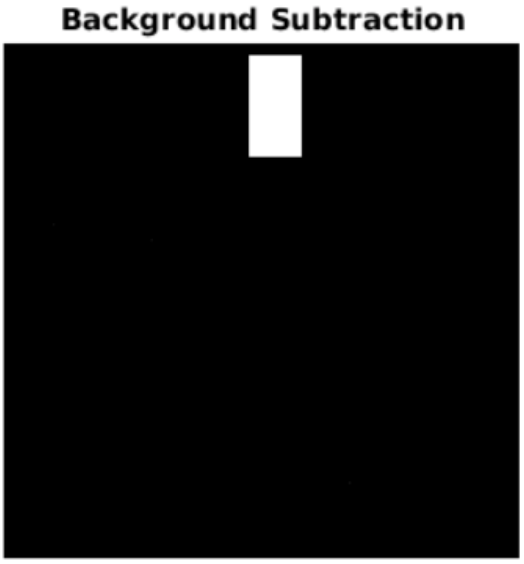


Image with Background removed

The above image represents the background image mask. As we can see, the result is quite fantastic.