Name: Bhavya Samhitha Mallineni
USC ID: 6580252371
USC Email: mallinen@usc.edu
Submission Date: 02/19/2024

REPORT
HOMEWORK2

Problem 1: Edge Detection

a. Sobel Edge Detector:

I. Abstract and Motivation :

A basic task in image processing is edge detection, which finds sudden variations in pixel intensity inside an image to define object borders. What are known as these sudden shifts, or discontinuities, are important markers of edges. The transitions between states are highlighted by edge detection approaches, which use mathematical processes like convolution with gradient operators.
The Sobel operator is a well-known technique among the many edge detection methods. In order to work, the Sobel operator convolves the image using specified kernels, which are basically tiny matrices that highlight gradients in both vertical and horizontal intensity. Sobel highlights areas in the image where there are notable fluctuations in pixel intensity on both the horizontal and vertical axes by applying first-order differentiations. The Sobel operator efficiently increases the gradient magnitude at points where edges are present by this process, giving a clear picture of object boundaries. This method helps separate important features from background clutter.

II. Approach and Procedure:

The tiger and pig color photos in this task are provided to us; to do edge detection, we first convert them to a grayscale image and follow the edge detection procedures..The Sobel operator consists of two filters: one for the x-axis and one for the y-axis. The 'x' filter represents the x-derivative, while the 'y' filter represents the y-derivative. To process our two images, we convolve each image with the x and y gradient filters separately, resulting in row and column derivatives. Subsequently, we normalize these derivative values. Next, we compute the magnitude using the x and y gradients and normalize the resulting magnitude image.To determine the threshold value for our probability map, we calculate the cumulative distribution function (CDF) of the normalized image. We select the threshold value as the point where the cumulative probability reaches 90%, aiming to enhance the visibility of borders in the resulting image.

-1	0	+1
-2	0	+2
-1	0	+1

Gx

+1	+2	+1
0	0	0
-1	-2	-1

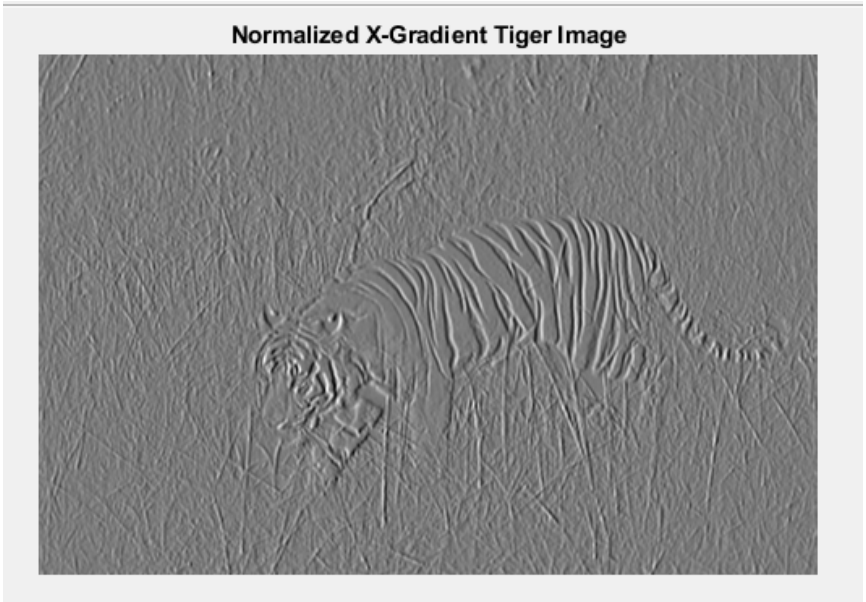
Gy

Picture 1: Sobel X and Y gradient filters

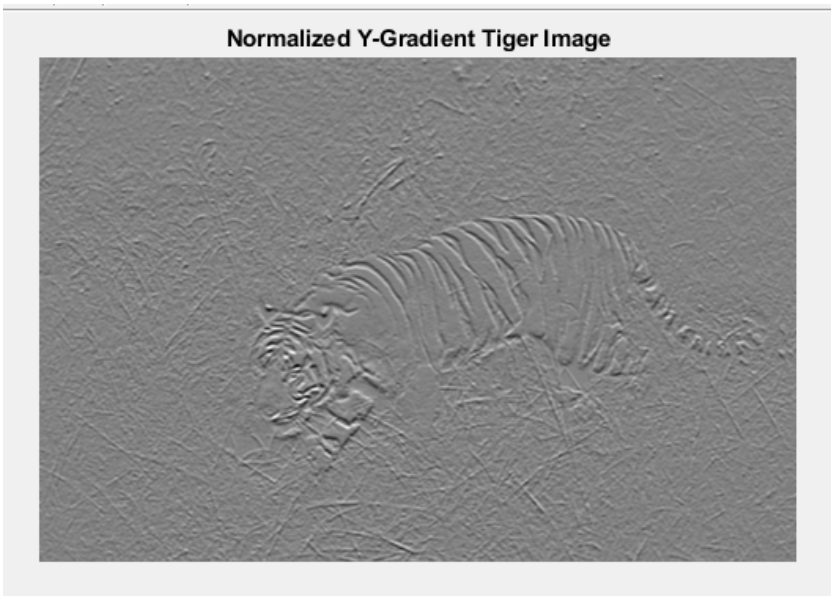
III. Experimental Results

Picture 2. Original Tiger Image

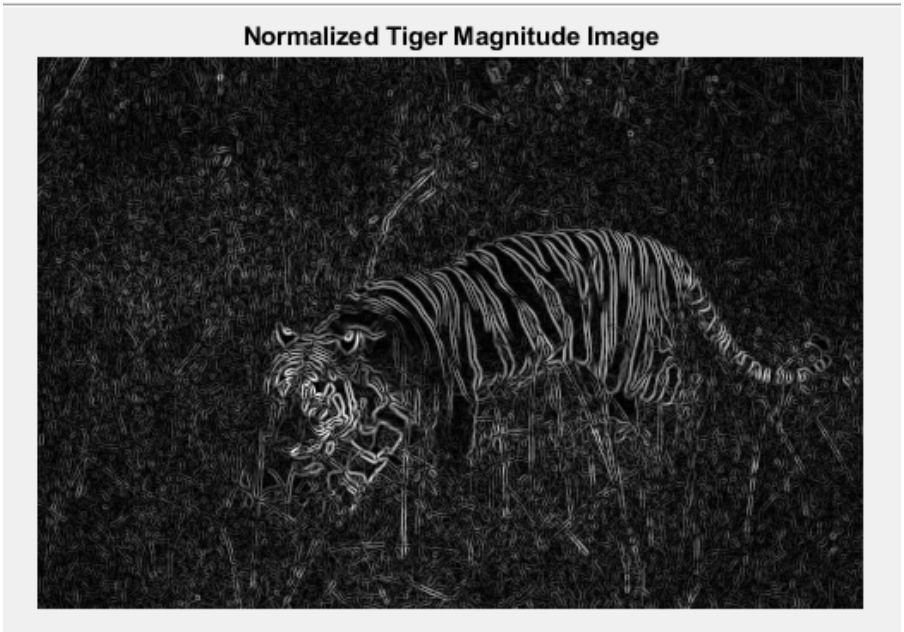
Picture 3: Grayscale tiger image



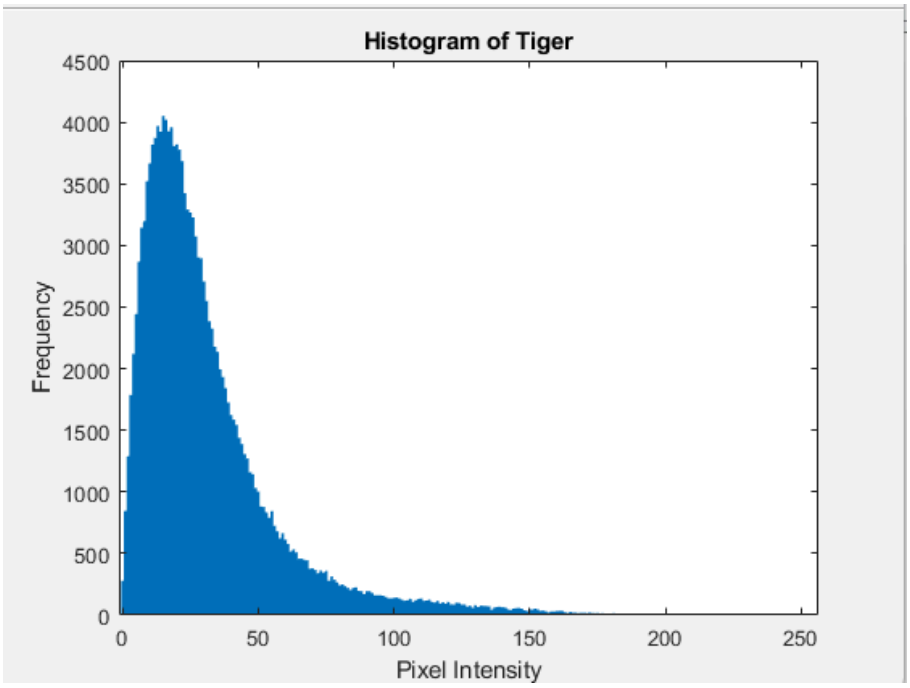
Picture 4: Normalized X-Gradient Tiger Image



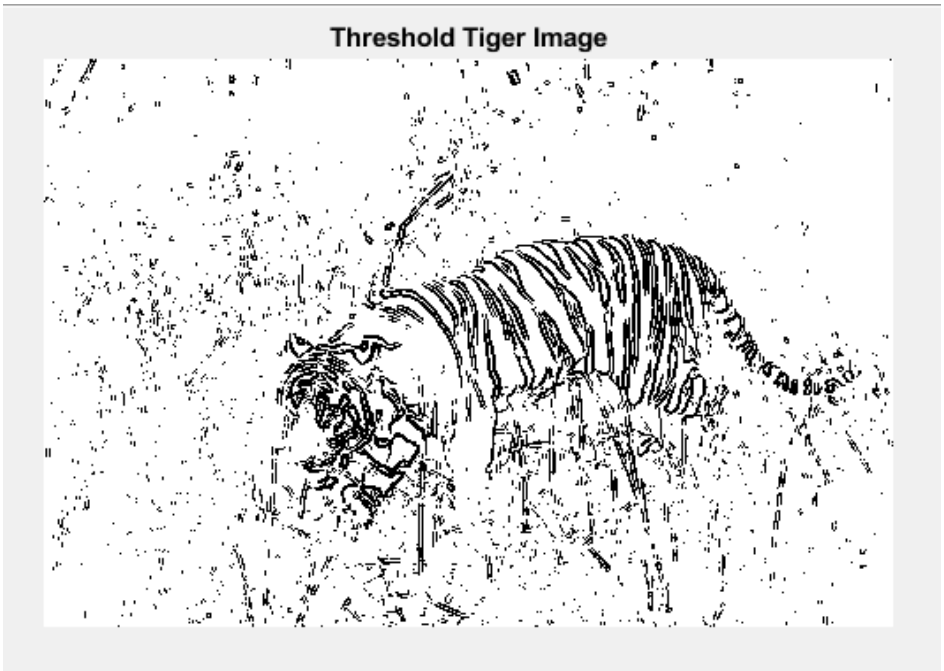
Picture 5: Normalized Y- Gradient Tiger Image



Picture 6: Normalized Probability Map of Tiger



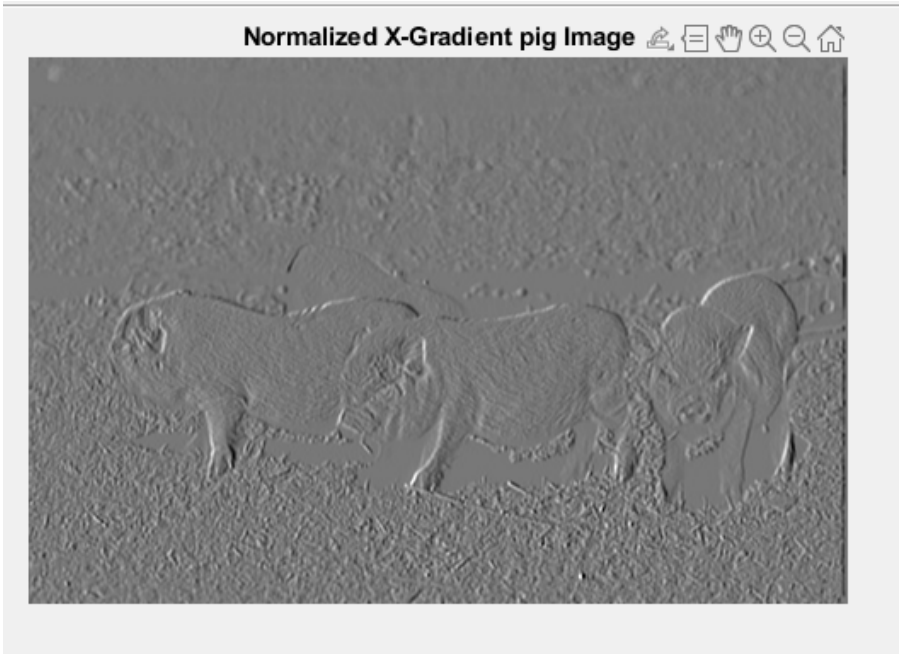
Picture7: Histogram showing the pixel distribution of probability map fo Tiger



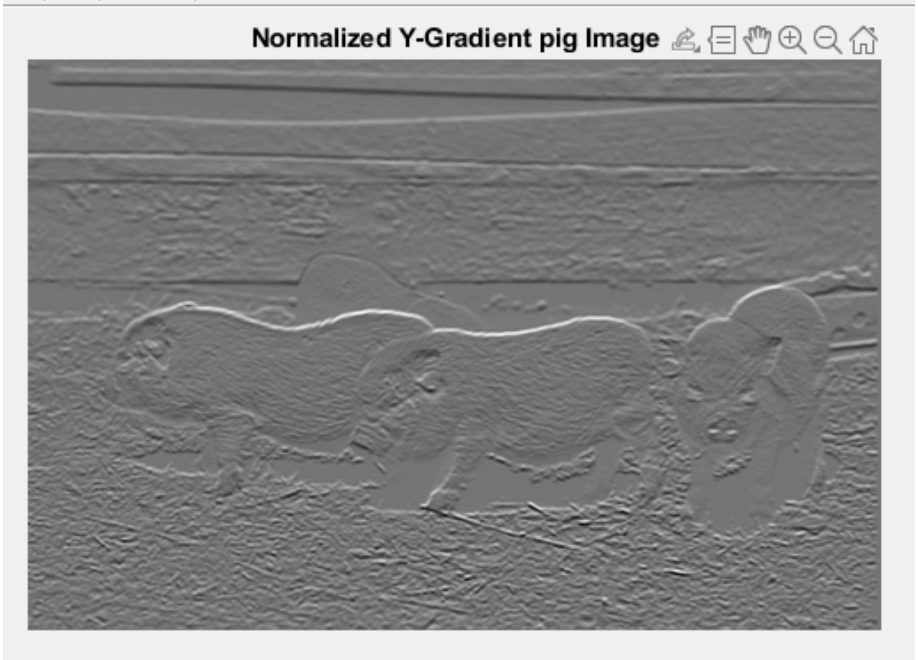
Picture 8: Threshold 90% = 63, 255 if < 63 and 0 if >63

Picture 9: Original Pig colour image

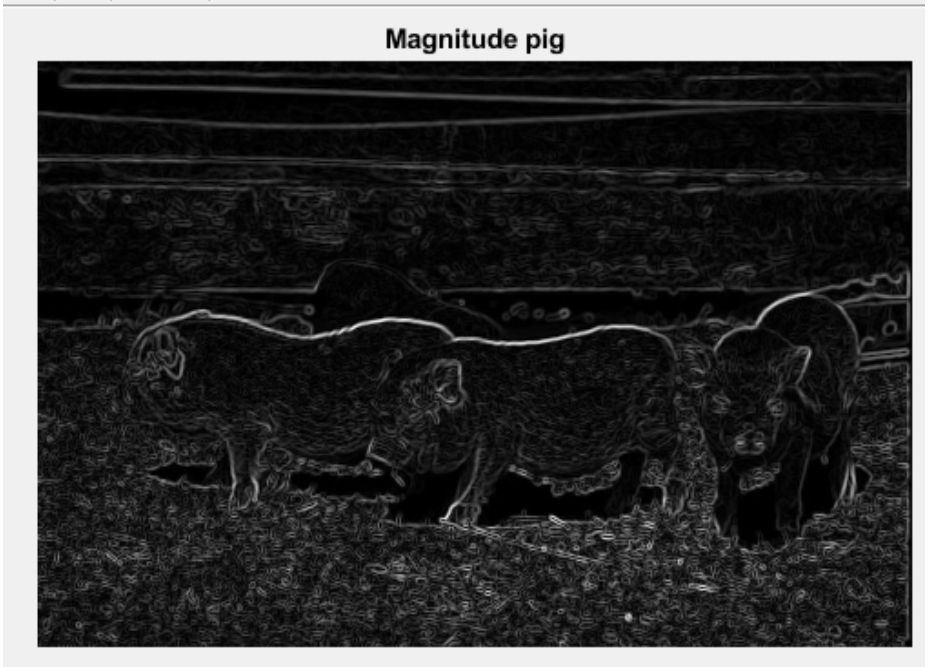
Picture 10: Grayscale Image of Pig



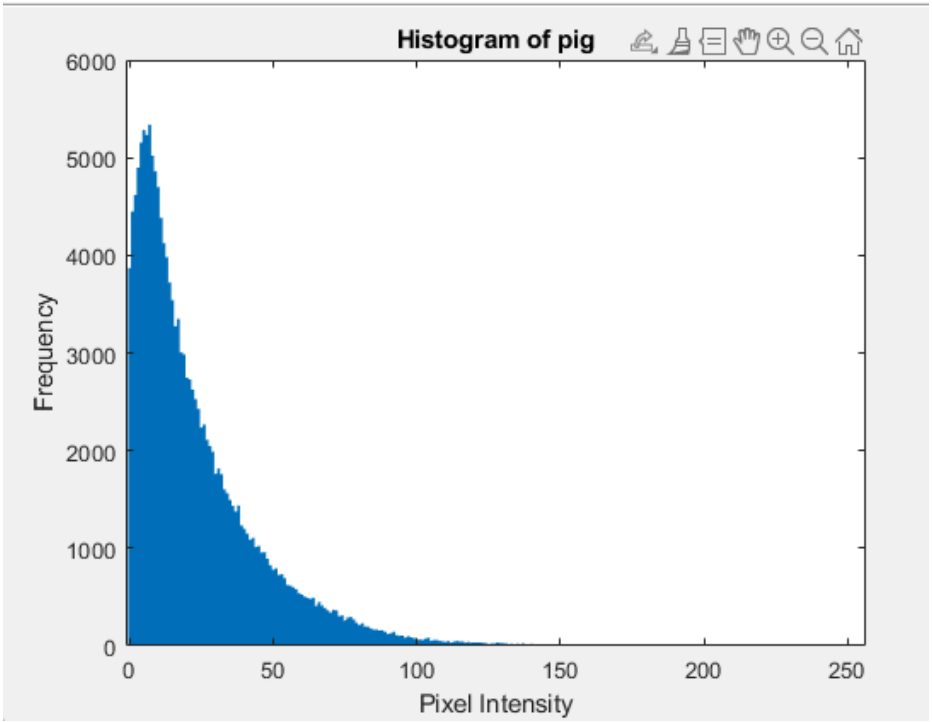
Picture 11: Normalized X gradient Pig Image



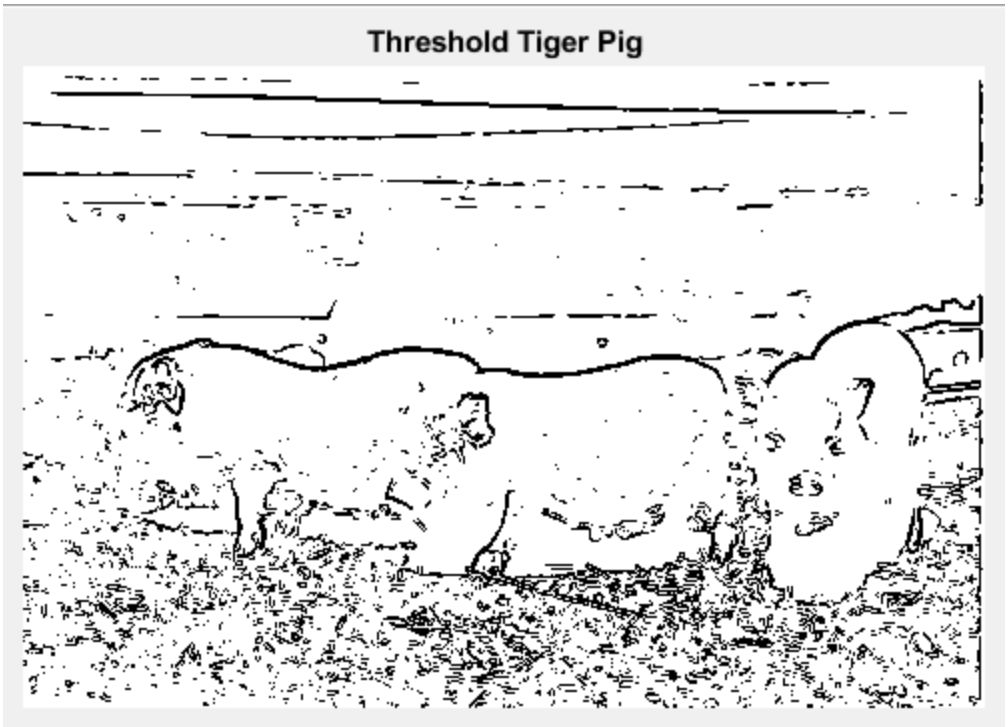
Picture 12: Normalized Y Gradient Pig Image



Picture 13: Normalized probability map pig image



Picture14: Histogram showing the pixel distribution of probability map fo Pig



Picture15: Threshold 90% = 55, 255 if < 55 and 0 if >55

IV. Discussion

After performing the Sobel edge detection, a threshold of 63 was applied for the tiger image, while for the pig image, it was set at 55. The edges were assigned a black color, and the background was made white to enhance edge visibility. While the edges of the animals' bodies were correctly identified, significant noise is present, which compromises the clarity of the image.

Further filtering techniques can be used to reduce the noise and enhance the clarity of the image. Methods that assist smooth out the noise while keeping the edges intact include median filtering and Gaussian blurring. Reducing noise while keeping crucial edge information may also be accomplished by adjusting the threshold value. Furthermore, you might improve the image quality and outcomes by adjusting the edge detection algorithm's settings or taking into account different edge detection techniques. To find the ideal balance between noise reduction and edge recognition accuracy, it is frequently required to experiment with various strategies and parameter

b. Canny Edge Detector

V. Abstract and Motivation

Canny edge detection is a refined technique inspired by the Sobel operator. It enhances the Sobel output by mitigating noise inherent in the gradient computations. By employing a combination of second-order derivatives and dual thresholds, Canny edge detection identifies edges more effectively. The process begins with computing the gradient

magnitude and direction across the image, highlighting regions of significant intensity change. However, to minimize the impact of noise, Canny applies Gaussian smoothing before gradient calculation. It then suppresses non-maximum pixels in the gradient direction to identify possible edges. Strong and weak edge pixels can be chosen due to the dual thresholds, which make it easier to trace continuous edges while ignoring weak or noisy signals. With less sensitivity to noise, Canny's multi-step method produces accurate edge maps that are essential for a variety of image processing applications.

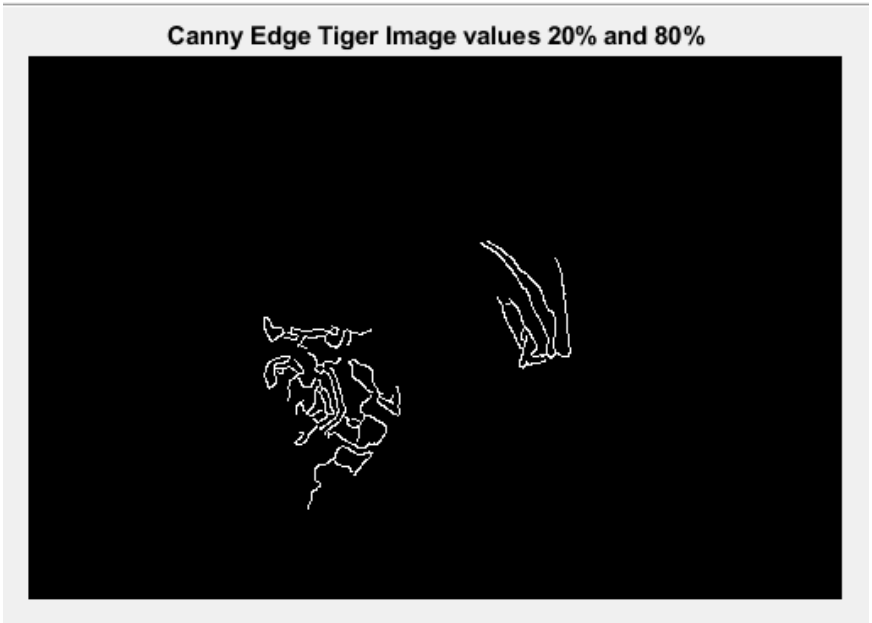
1. **Non- Maximum Suppression:-**
   In the Canny edge detection technique, non-maximum suppression plays a crucial role in identifying the most distinct edges within an image. Each pixel in the image is assigned a gradient magnitude and orientation, representing the direction and strength of the edge at that point. During non-maximum suppression, every pixel is compared with its neighboring pixels along the direction of the edge. If a pixel possesses the highest magnitude among its neighbors, it's considered a potential edge point and retained. However, if its magnitude is not the greatest, it's likely not a significant edge, so its intensity is reduced to zero, effectively suppressing it from further consideration as an edge pixel. This selective process ensures that only the most prominent edges are retained, enhancing the accuracy of edge detection. This gives an advantage over Sobel operator as it can generate edges that are thinner and more precise.

2. **Dual Thresholding :-**
   Double thresholding is a critical stage in the Canny edge detection algorithm, utilizing both high and low thresholds to classify pixels based on their gradient magnitudes. Pixels surpassing the high threshold are labeled as strong edges, denoting pronounced intensity shifts within the image. These strong edges typically signify substantial features or boundaries. Pixels with gradient magnitudes falling between the low and high thresholds are identified as weak edges. While weak edges may correspond to genuine edges, they could also result from noise or less pronounced features, necessitating further validation. Conversely, pixels below the low threshold are deemed non-edges and are disregarded from the edge map. By employing dual thresholds, the Canny algorithm enhances the discrimination between true edges and noise, fostering the creation of a refined edge map that accurately delineates significant transitions in intensity across the image.

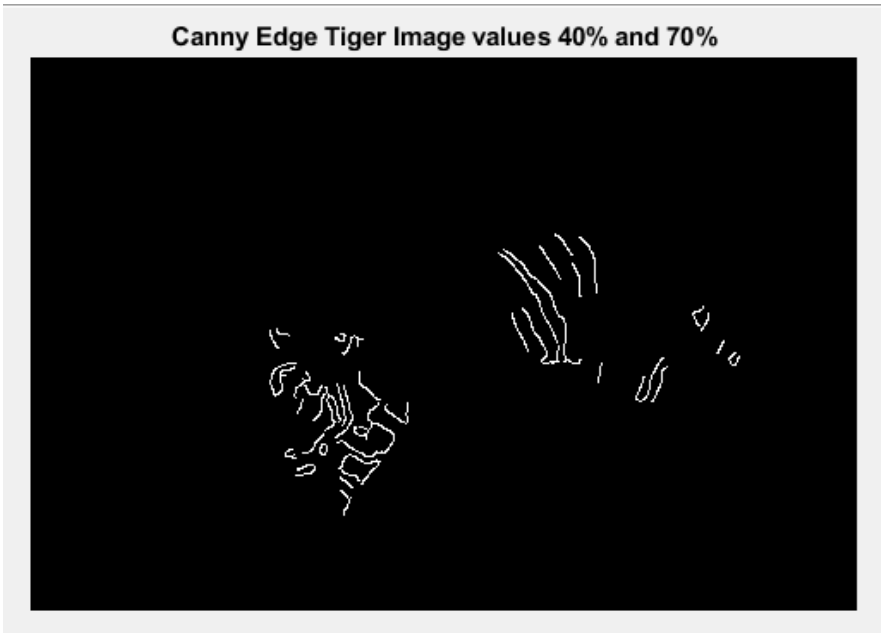
VI. Approach and Procedure

For our task, we rely on MATLAB's "edge.m" function, which facilitates various edge detection techniques. We are using the canny edge detection, in which initially, gaussian smoothing minimizes noise, preparing the image for analysis. Next, the Sobel operator computes gradients, highlighting potential edges. Non-maximum suppression refines the edges, retaining local maxima. Double thresholding categorizes pixels into strong, weak, or non-edges, resulting in a nuanced edge map.

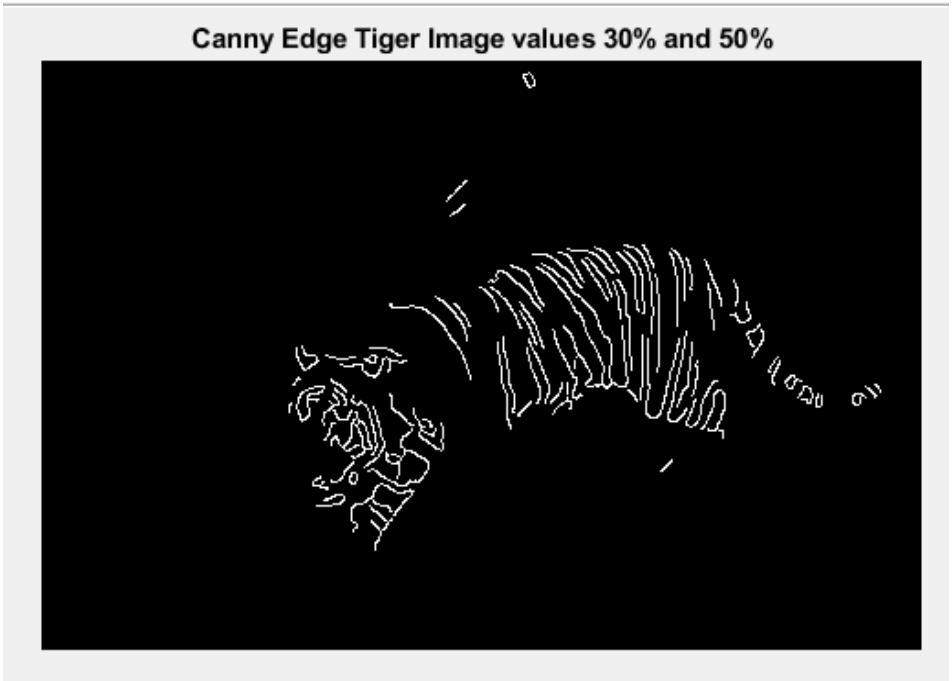
VII. Experimental Results



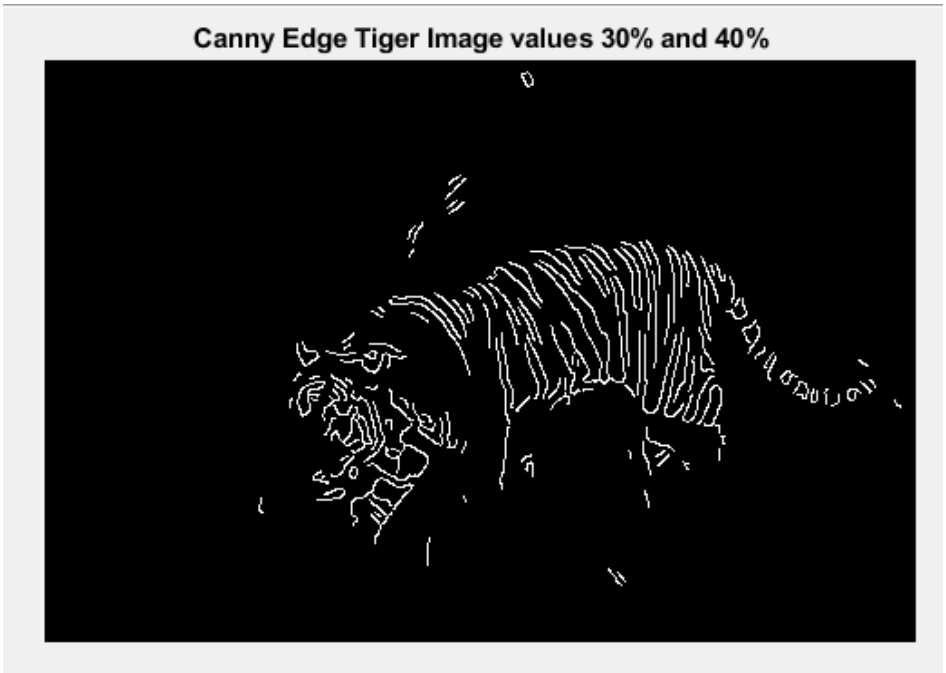
Picture 16: taking threshold levels 20% and 80% giving a bad outcome



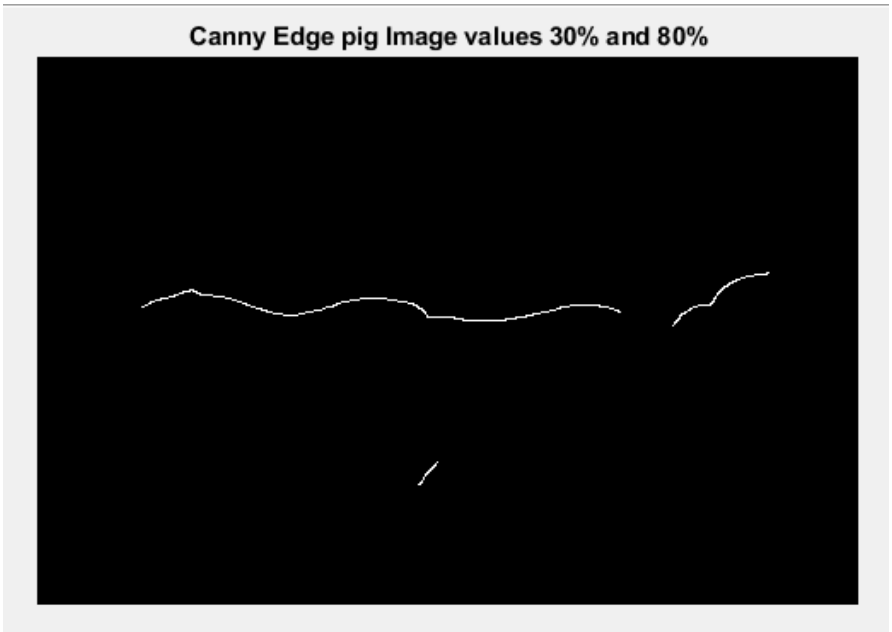
Picture 17: taking threshold levels 40% and 70% giving a bad outcome



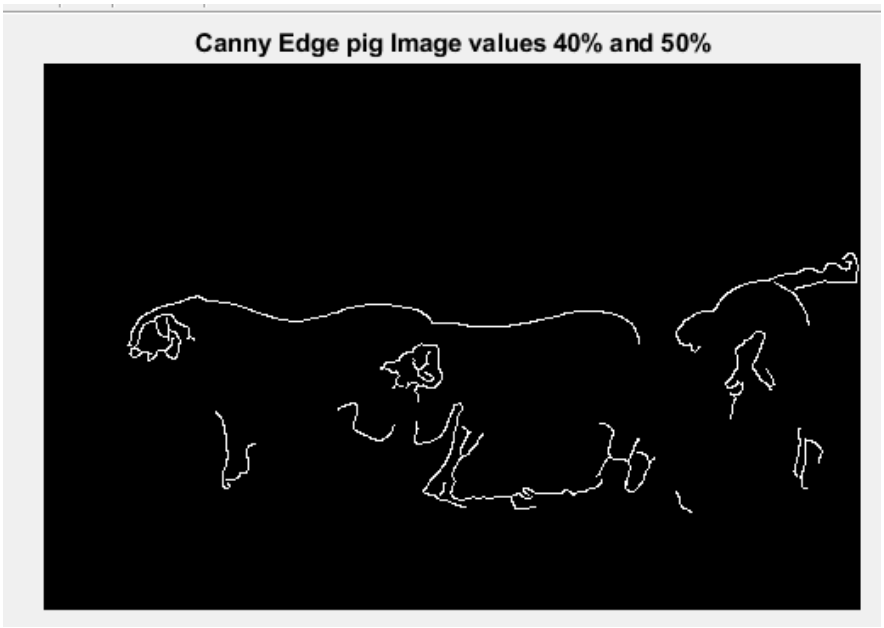
Picture 18: taking threshold levels 30% and 50% giving a better outcome



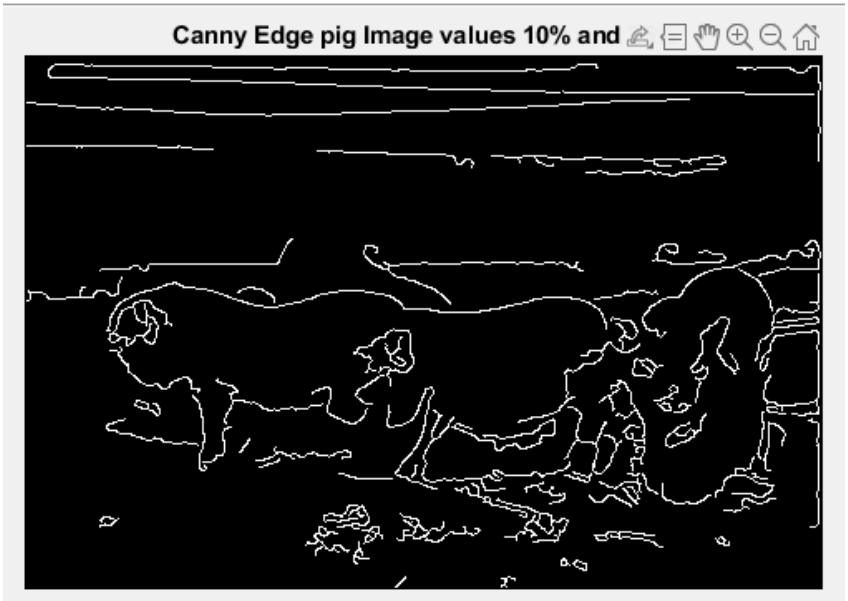
Picture 19: taking threshold levels 30% and 40% giving a better outcome but a little noise



Picture 20: taking threshold levels 30% and 80% giving a bad outcome



Picture 21: taking threshold levels 40% and 50% giving a better outcome



Picture 22: taking threshold levels 10% and 35% giving a better outcome but a lot of noise

VIII. Discussion

While experimenting with Canny edge detection on images of a tiger, we found that using threshold levels of 20% and 80% resulted in poor outcomes. However, employing threshold levels of 30% and 50% yielded better results. Additionally, using threshold levels of 30% and 40% produced improved outcomes, albeit with some noise.
Similarly, when applying Canny edge detection to images of a pig, using threshold levels of 30% and 80% led to unsatisfactory results. However, employing threshold levels of 40% and 50% produced better outcomes. Furthermore, using threshold levels of 10% and 35% resulted in improved outcomes, despite significant noise.
The main observation is that when the upper threshold limit is higher, only the strongest edges are considered, which may be few in number. As we decrease the threshold limits, weaker edges are also included, but this may result in more noise being detected as edges.

c. Structured Edge Detector

IX. Abstract and Motivation

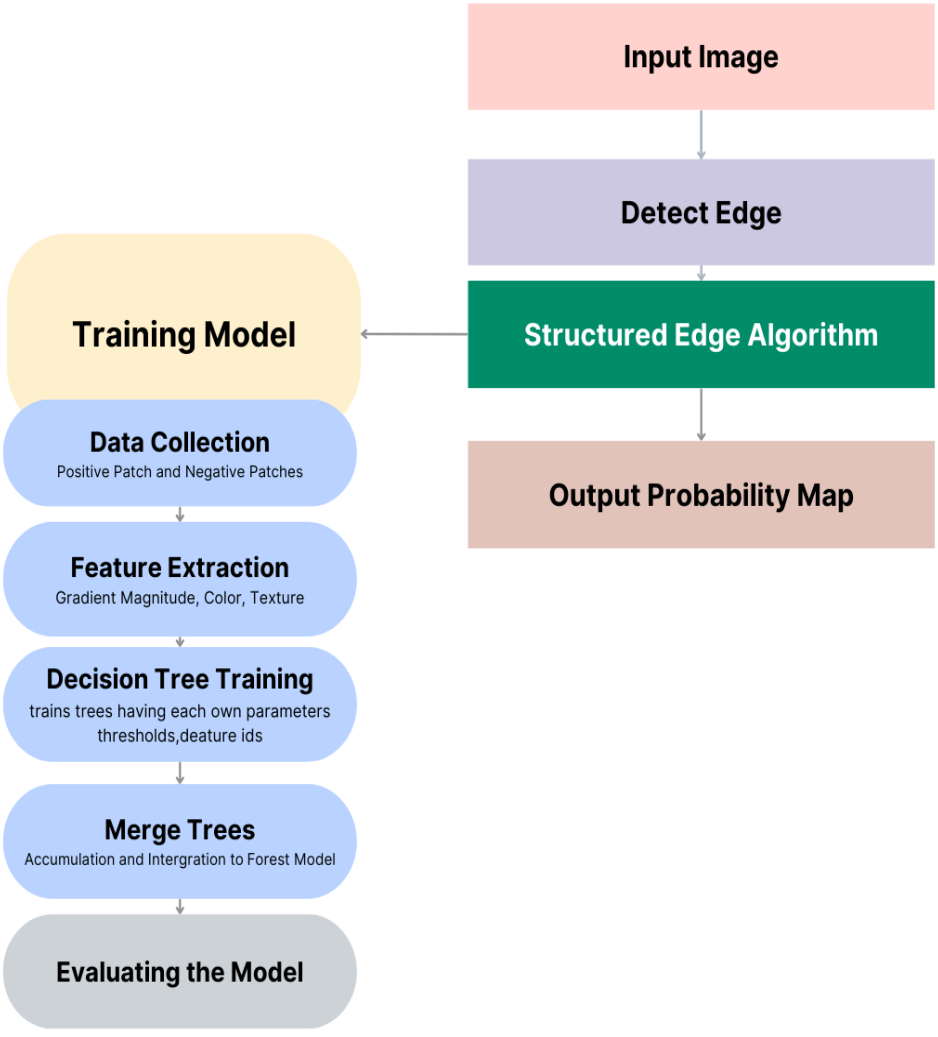
The Structured Edge Detection technique employs machine learning to identify edges in images, distinguishing itself from traditional methods like Sobel and Canny edge detectors. Rather than directly applying filters to detect edges, Structured Edge Detection relies on a pre-trained model specifically designed to recognize edges within

images. A key aspect of this technique involves utilizing the Random Forest method for model learning. Random Forest is a powerful ensemble learning algorithm that constructs a multitude of decision trees during training and combines their outputs to make predictions. In the context of Structured Edge Detection, Random Forest learns to recognize patterns indicative of edges based on features extracted from image patches. By leveraging machine learning and Random Forest, Structured Edge Detection achieves robust edge detection performance, particularly in scenarios with complex textures and varying image conditions.

X. Approach and Procedure

We are using the Structured Edge Toolbox in MATLAB to perform edge detection.

1. Structured Edge Algorithm Flowchart:



When an image is provided for edge detection in a structured algorithm, it calls upon its pretrained model for use. The pretrained model is constructed using a dataset of images containing ground truths and thresholds. These images serve as training data. Positive and negative patches are gathered from these training images. Positive patches generally encompass edges or regions of interest, while negative patches represent background or non-edge areas. Positive patches are usually extracted around ground truth edge locations, while negative patches are selected randomly.

Features are computed for each patch, which commonly include gradient magnitude, color, texture, and other traits crucial for distinguishing between edges and non-edges. These features are often assessed across different scales and orientations to capture a variety of edge characteristics. Each tree is independently trained using these features and their corresponding positive and negative labels.

The decision tree learning algorithm recursively splits the feature space based on feature values, aiming to reduce impurity, while maximizing information gain. Training persists until specific stopping criteria, such as reaching a maximum depth or minimum number of samples per leaf node, are met. Following the training of all trees, they are combined into a single model structure. Each tree retains its own parameters, including threshold values, feature IDs.

Model Integration merges the individual trees into the final model structure, consolidating parameters like thresholds, feature IDs, child indices, and node counts. This unified model represents a forest of decision trees trained to identify edges in images. Each tree contributes to the overall decision-making process during edge detection. The finalized model is primed for use in edge detection tasks on new images. Upon processing the input image through the model, it yields parameters and predicted labels. These predicted labels enable the determination of whether the image contains an edge or not.

2. **Random Forest Classifier:**

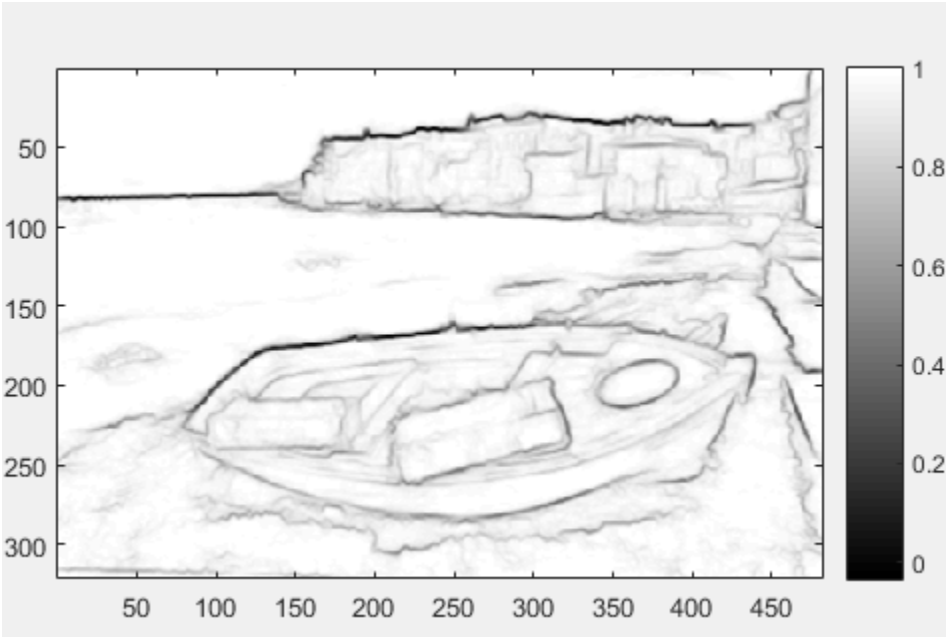
A decision tree is a machine-learning tool used for both classification and regression tasks. It constructs a model represented as a tree with decision nodes and leaf nodes, where decisions are made sequentially to achieve a specific outcome. However, using a single decision tree may not always yield effective results. This is where the Random Forest algorithm comes in. Random Forest is a type of machine learning algorithm that relies on multiple decision trees to make predictions. In a random forest, each decision tree operates on a random subset of features to predict outcomes. These individual tree predictions are then combined to produce the final output. Bootstrapping, a technique of randomly selecting items from a dataset, is employed in the process. This technique gathers randomized decisions from multiple trees and uses majority voting to make the final decision.

In our structured edge model, the decision trees are constructed iteratively. At each node, the algorithm selects the optimal feature based on information gain. This iterative process halts when the maximum depth or the minimum samples per leaf node are met. The forest integrates the decision boundary learned by each tree. During inference, the

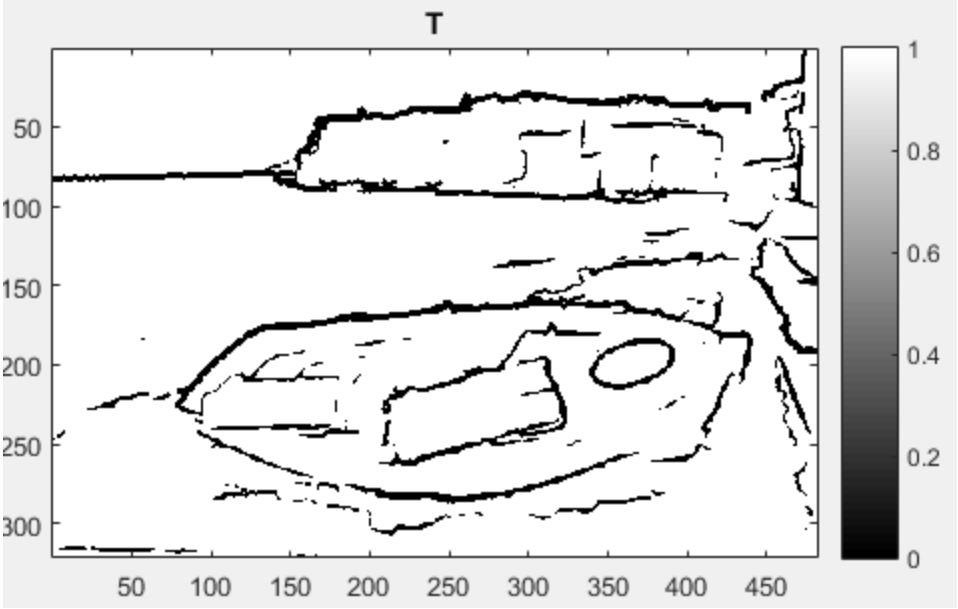
forest evaluates multiple trees to collectively determine edge presence, improving edge detection accuracy and resilience to noise or variations in image content.

XI. Experimental Results

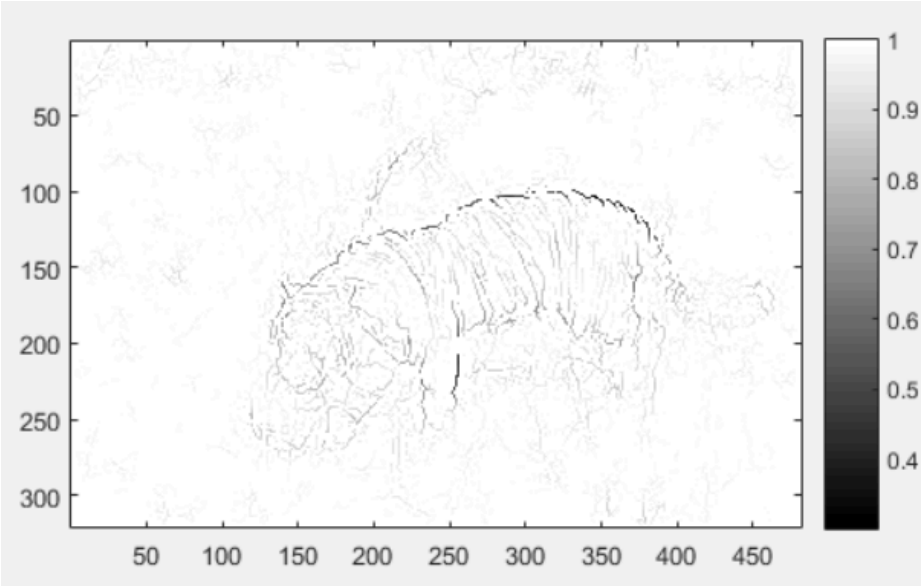
Picture 23: Boat Original Image



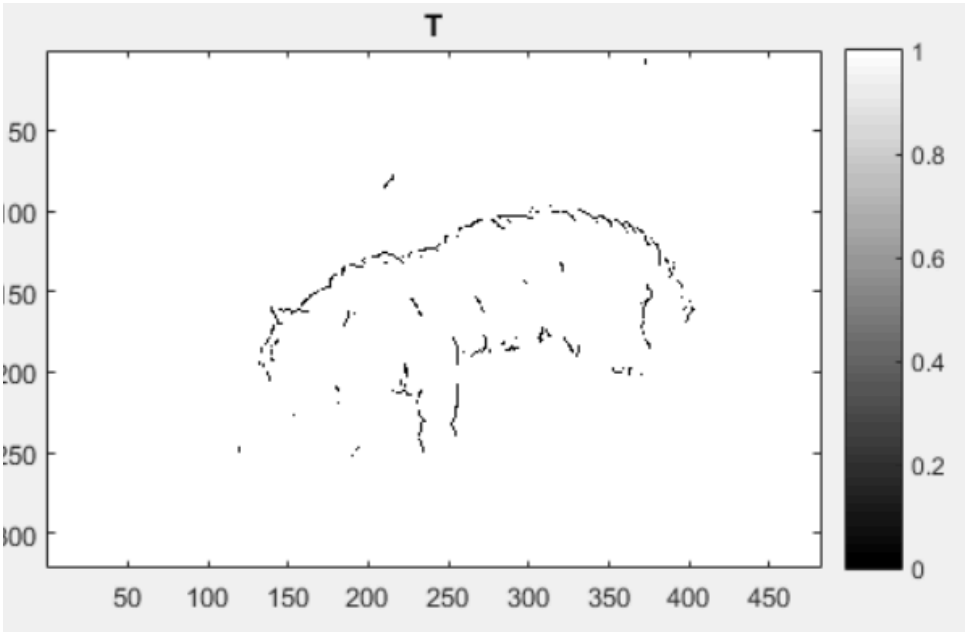
Picture 24: Boat Probability Edge Map



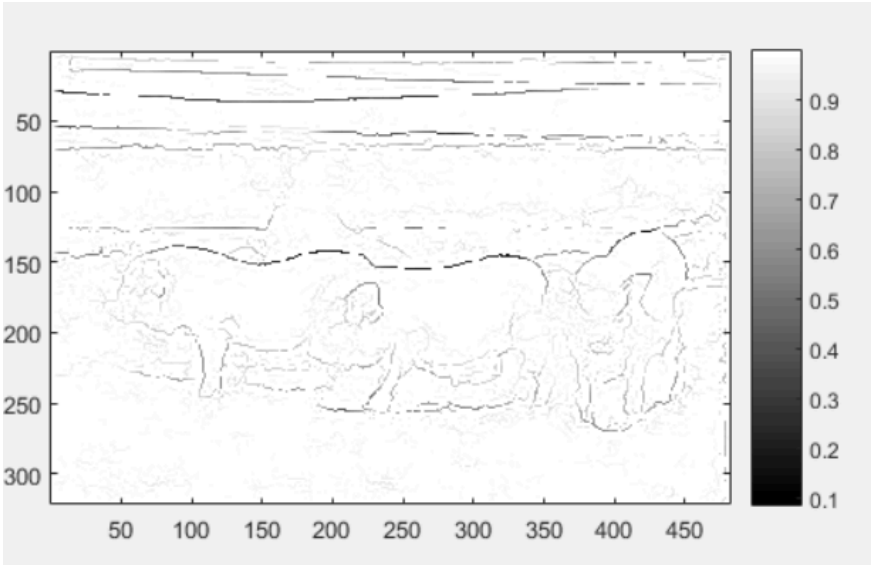
Picture 25: Boat Threshold Edge Map



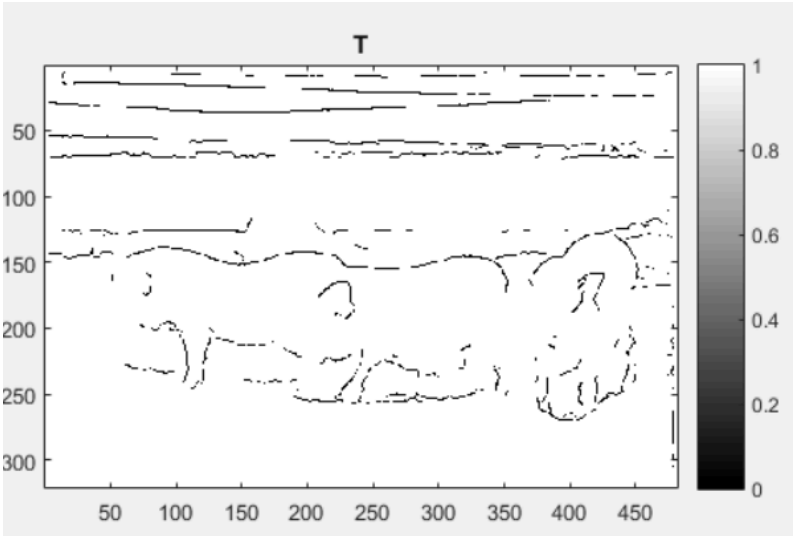
Picture 26: Tiger Probability Edge Map



Picture 27: Tiger Threshold Edge Map



Picture 28: Pig Probability Edge Map



Picture 29: Pig Threshold Edge Map

XII. Discussion

The tiger and pig images underwent structured edge detection with non-maximum suppression (NMS) enabled, akin to Canny edge detection. Setting the threshold to 0.8 significantly reduced noise compared to lower thresholds. Structured edge detection presents less noise compared to Canny and Sobel methods. By emphasizing prominent edges and suppressing weaker ones, it enhances edge clarity, yielding cleaner images. This approach minimizes artifacts and irrelevant details, resulting in visually appealing edge maps. Consequently, structured edge detection proves advantageous in scenarios requiring precise edge delineation and minimal noise interference, offering superior image quality and facilitating clearer visual analysis.

d. Performance Evaluation.

XIII. Abstract and Motivation

Performance evaluation metrics such as precision, recall, accuracy, and F1 score are crucial for assessing the effectiveness of machine learning models.Precision measures the proportion of correctly identified positive cases out of all instances identified as positive, emphasizing the model's ability to avoid false positives.Recall, also known as sensitivity, gauges the model's capacity to capture all positive instances, representing the proportion of true positives identified correctly out of all actual positives.Accuracy measures the overall correctness of the model by calculating the ratio of correctly predicted instances to the total instances evaluated.F1 score, the harmonic mean of precision and recall, offers a balanced assessment of the model's performance, considering both false positives and false negatives. It is particularly useful when dealing with imbalanced datasets.

These metrics collectively provide insights into different aspects of model performance, aiding in informed decision-making and model optimization.

XIV. Approach and Procedure.

For our metrics calculation, we are utilizing the edgesEvalImg() function in the SE toolkit. This function evaluates edge precision and recall results for a single edge image against ground truth boundaries. It takes as input an edge probability map E and ground truth boundaries G, along with optional parameters such as thresholds and maximum distance for edge matching. The function computes precision and recall ratios at different thresholds and can create visualizations of edge matches. Additionally, the results can be optionally written to an output file.

$$
\begin{aligned}
precision &= \frac{TP}{TP + FP} \\
recall &= \frac{TP}{TP + FN} \\
F1 &= \frac{2 \times precision \times recall}{precision + recall}
\end{aligned}
$$

XV. Experimental Results

Evaluation for Tiger:

Sobel Mean Precision: 0.14707
Canny Mean Precision: 0.33117
SE Mean Precision: 0.21719

Filter	GT1	GT2	GT3	GT4	GT5
Sobel	0.0861	0.0870	0.0895	0.3719	0.1009
Canny	0.1539	0.1730	0.1803	0.9549	0.1937
Structured Edge	0.1258	0.1287	0.1517	0.5518	0.1279

Sobel Mean Recall: 0.31839
Canny mean Recall: 0.38976
SE Mean Recall:0.5813

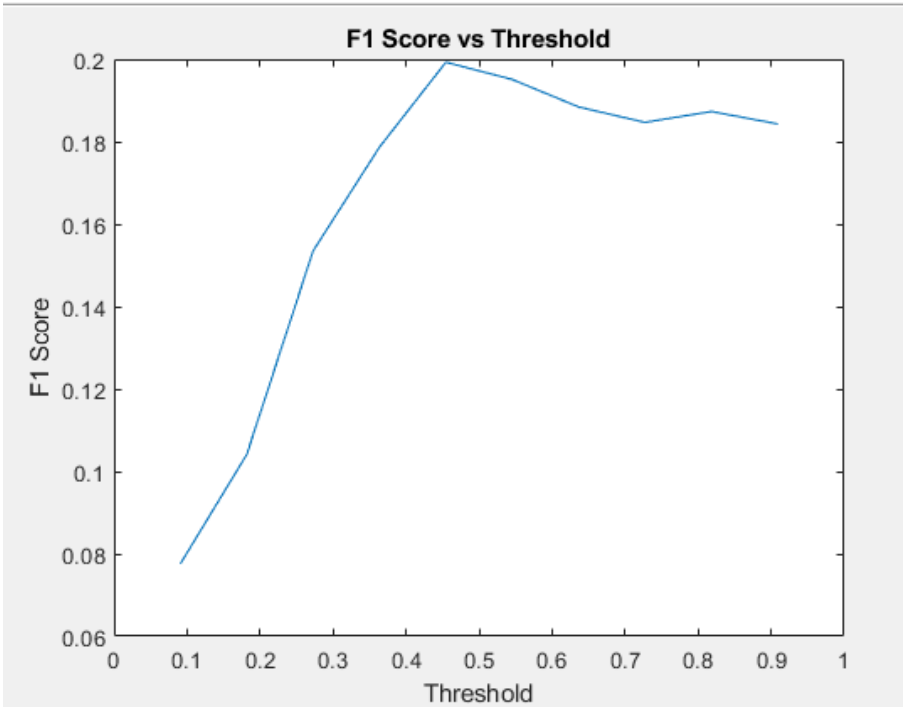
Filter	GT1	GT2	GT3	GT4	GT5
Sobel	0.3465	0.3302	0.3025	0.3188	0.2940

Canny	0.3694	0.3915	0.3634	0.4881	0.3364
Structured Edge	0.2805	0.2705	0.2841	0.2620	0.2064

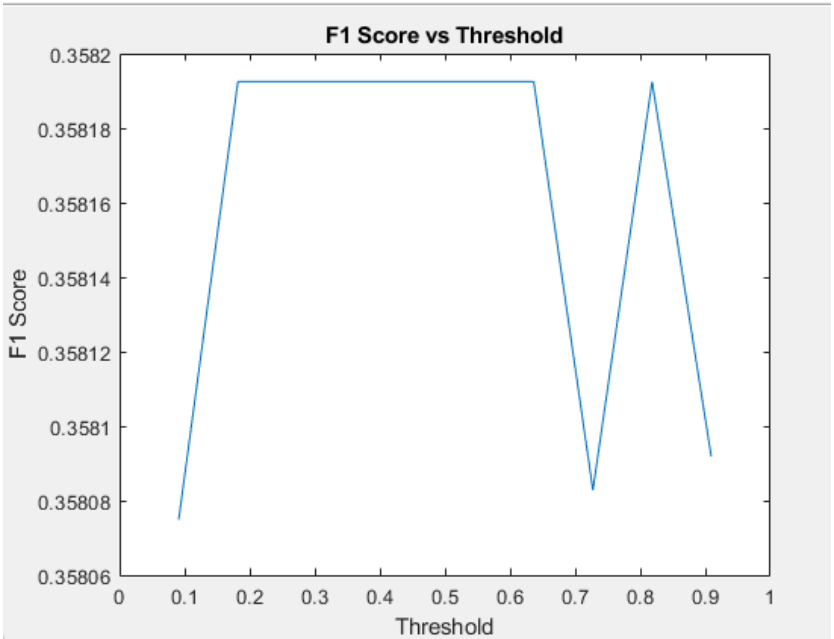
F1 Score:

Filter	**F1 Score**
Sobel	0.20121
Canny	0.35808
Structured Edge	0.23696

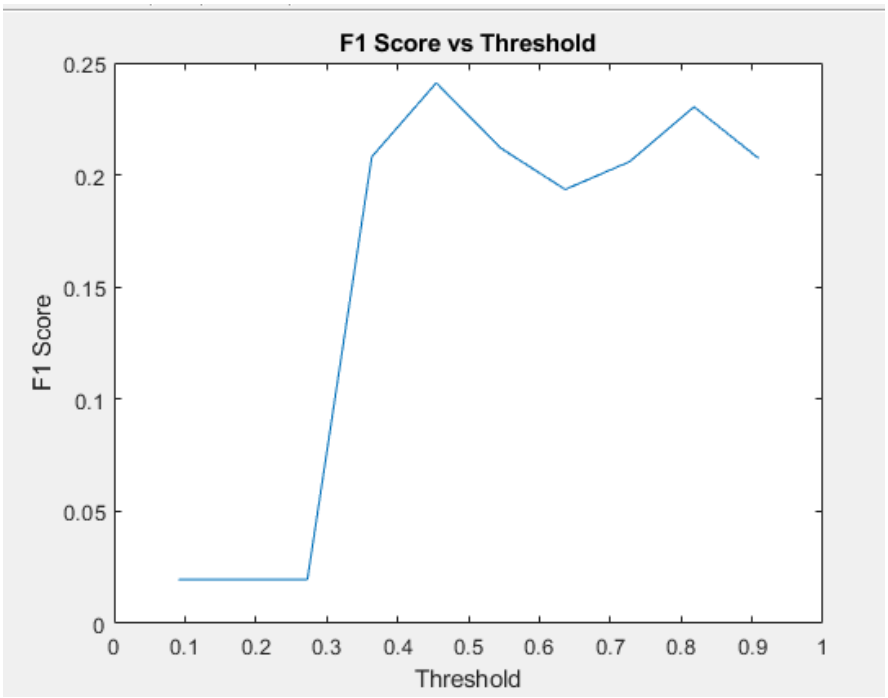
Sobel:



Canny:



Structured Edge:



Evaluation for Pig:

Sobel Mean Precision:0.16966
Canny Mean Precision: 0.34831
SE Mean Precision: 0.21784

Filter	GT1	GT2	GT3	GT4	GT5
Sobel	0.1141	0.1216	0.1880	0.2395	0.1851
Canny	0.2448	0.2636	0.3606	0.5104	0.3621
Structured Edge	0.1329	0.1509	0.2332	0.3057	0.2666

Sobel Mean Recall:0.52313
Canny mean Recall: 0.70493
SE Mean Recall:0.25296

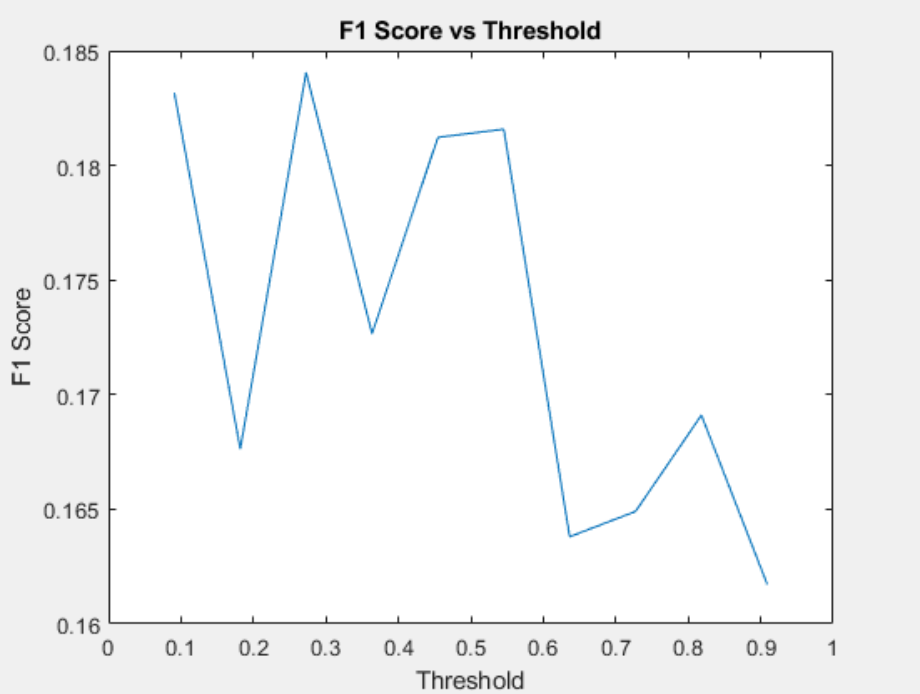
Filter	GT1	GT2	GT3	GT4	GT5
Sobel	0.5379	0.5302	0.5120	0.5191	0.5164
Canny	0.7530	0.7501	0.6407	0.7217	0.6591
Structured Edge	0.2383	0.2503	0.2415	0.2519	0.2828

F1 Score:

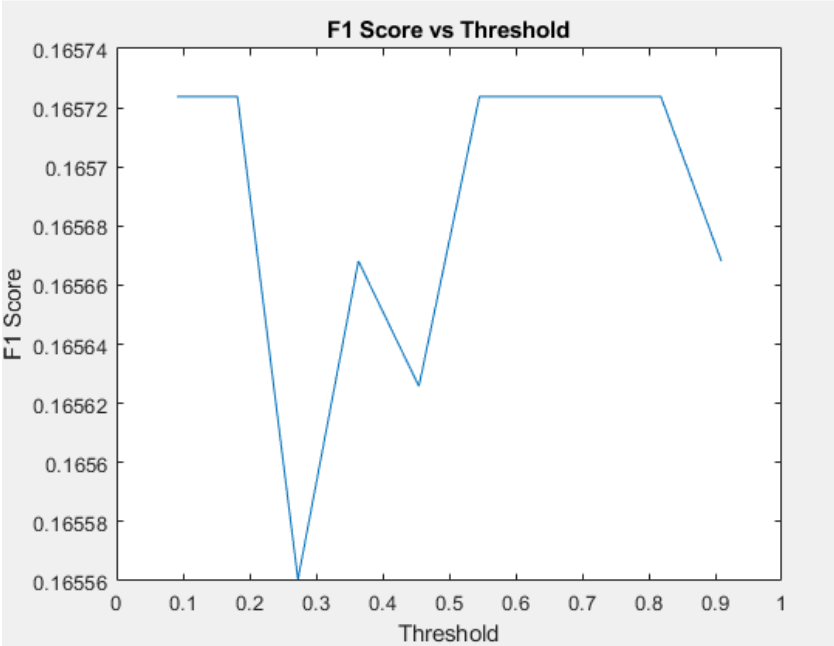
Filter	F1 Score
Sobel	0.25622
Canny	0.46637

Structured Edge	0.23409

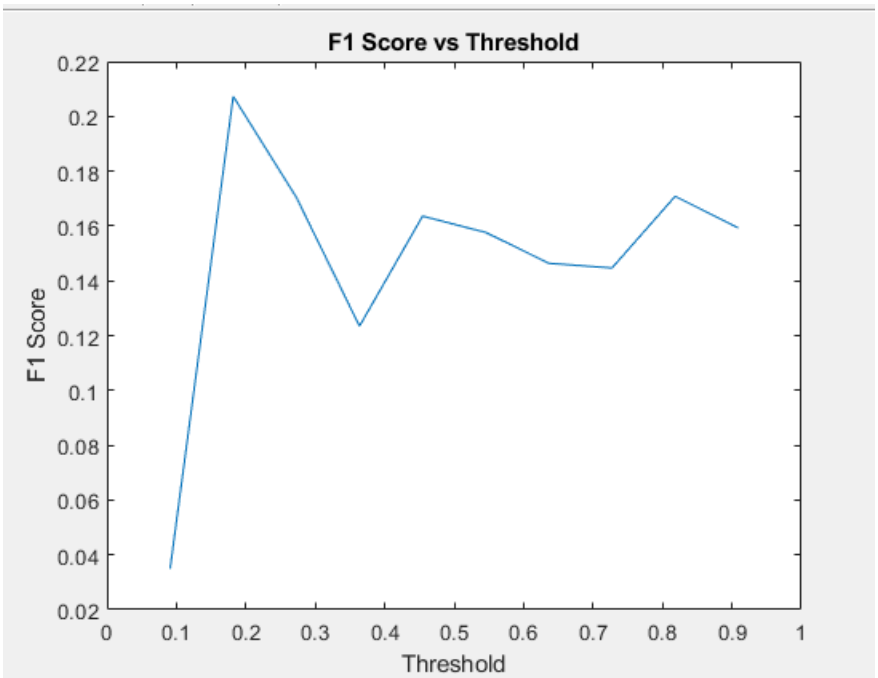
Sobel:



Canny:



Structured Edge:



XVI. Discussion:

1. Sobel prioritizes recall over precision, detecting numerous edges but also generating false positives. Conversely, Canny achieves a balance between precision and recall, rendering it suitable for general edge detection tasks. Structured Edge strikes a middle ground, offering moderate precision and recall. From all the values, it can be understood that the Canny edge detection method is best suited for our images.
2. We plotted the F1 scores for the three filters across three images at various thresholds. It's evident that Canny yields the highest F1 score in both images. Additionally, for Canny, these F1 scores fluctuate continuously. The next best performer is Structured Edge, which exhibits less fluctuation compared to Canny.
3. Comparing the graphs of both images, it can be said that it's easier for the pig image to achieve a higher F1 score. This assertion is supported by the observation that the recall and precision values are more balanced for the pig image compared to the tiger image. In the context of edge detection, a balanced trade-off between recall and precision is essential for achieving a high F1 score. The pig image seems to exhibit a better balance between the two metrics, indicating that the algorithm performs relatively consistently in capturing both true positives and avoiding false positives
4. Precision only indicates the accuracy of positive predictions, while recall assesses the coverage of positive instances, offering limited insights individually. Both metrics lack a holistic view of model performance, as optimizing one may compromise the other. Therefore, the F1 score is employed to strike a balance between precision and recall, providing a more comprehensive evaluation of classifier effectiveness, particularly in scenarios with imbalanced classes.
5. The F1 score's design revolves around harmonizing precision and recall, crucial metrics in classification tasks. By computing their harmonic mean, it ensures that both aspects are

equally weighted, thus offering a balanced evaluation of a model's performance. Small alterations in precision or recall can lead to noticeable shifts in the F1 score, reflecting the model's behavior accurately across different thresholds and scenarios. This sensitivity underscores the F1 score's effectiveness as a comprehensive metric, providing insights into both the accuracy of positive predictions and the coverage of positive instances, which are vital in assessing classifier performance.

F1= 2*P*R/(P+R)

Now lets assume P =R

F1= 2*P^2/2P, then F1 =P this shows F1 score reaches its maximum when there is a balance between precision and recall.

2a. Digital Halftoning

XVII. Abstract and Motivation

Digital halftoning is a technique used in printing and displaying grayscale images using only black ink or pixels of varying intensities to simulate shades of gray. It's employed primarily to replicate continuous-tone images with limited resources, like black-and-white printers or low-resolution displays.In digital halftoning, the image is broken down into a grid of dots or pixels. Instead of varying the intensity of each dot, the technique varies the size or spacing of dots to create the illusion of different shades of gray. This is achieved through algorithms that determine the arrangement of dots based on the desired grayscale levels and the capabilities of the output device.

XVIII. Approach and Procedure

a. **Fixed Thresholding :** a single threshold value is selected based on the desired segmentation of the image into foreground and background. Pixels with intensity values below this threshold are set to 0 (black), while those above are set to 255 (white). This method is straightforward but lacks adaptability to varying image characteristics.

$$b(i,j) = \begin{cases} 255 & \text{if } X(i,j) > T \\ 0 & \text{otherwise} \end{cases} .$$

b. **Random Thresholding:** Random thresholding involves the selection of threshold values through a random process. This technique introduces variability in the thresholding process, which can be advantageous in handling images with diverse content and lighting conditions. By randomly choosing thresholds, the

method can adapt to different image properties and improve segmentation results. *Here **T is rand(T)***

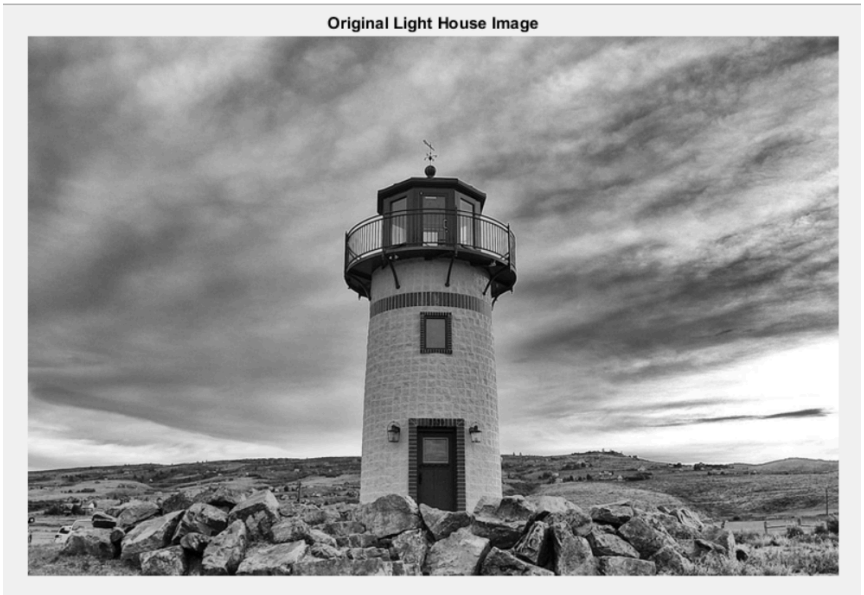
c. **Dithering Matrix:** Dithering matrices are used to create the illusion of additional colors and shades in an image by strategically placing patterns of black and white pixels. These patterns help simulate intermediate shades and gradients, producing a smoother transition between regions of different intensities. Dithering matrices are particularly useful in scenarios where the output medium has limited color depth or resolution, such as printing or displaying images on screens with fewer available colors.

$$I_2(i,j) = \begin{bmatrix} 1 & 2 \\ 3 & 0 \end{bmatrix}$$

$$T(i,j) = 255\frac{I(i,j)+0.5}{N^2}$$

$$b(i,j) = \begin{cases} 255 & \text{if } X(i,j) > T(i \bmod N, j \bmod N) \\ 0 & \text{otherwise} \end{cases}$$

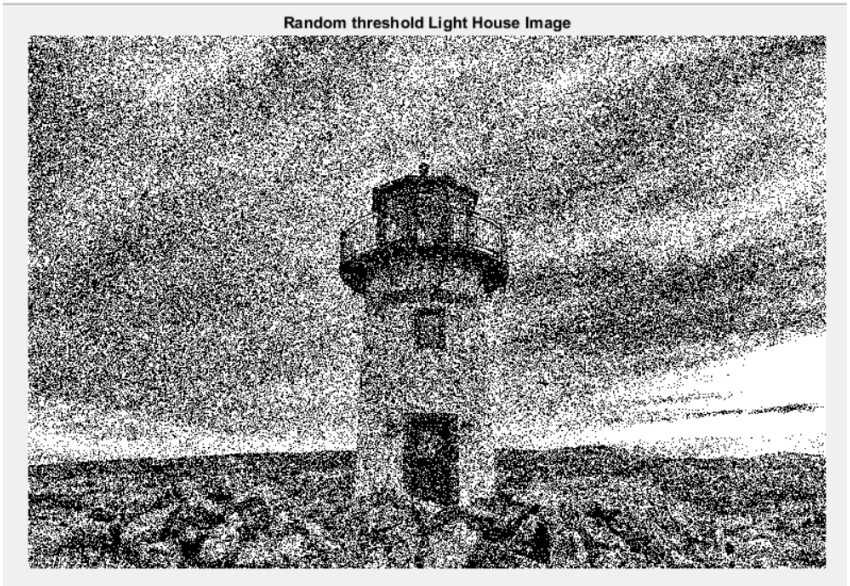
XIX. Experimental Results



Picture 30: Light House Original Image

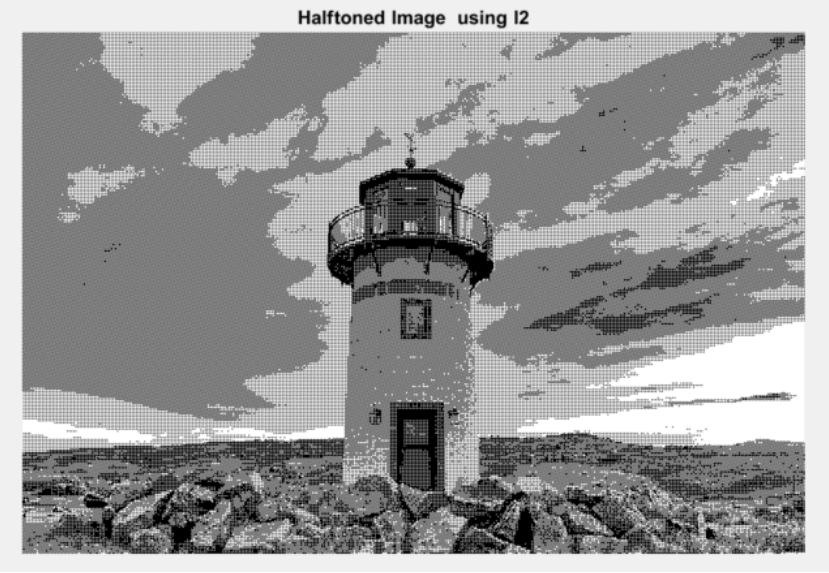
Picture 31: Fixed Thresholding Light House Image

2. Random Thresholding

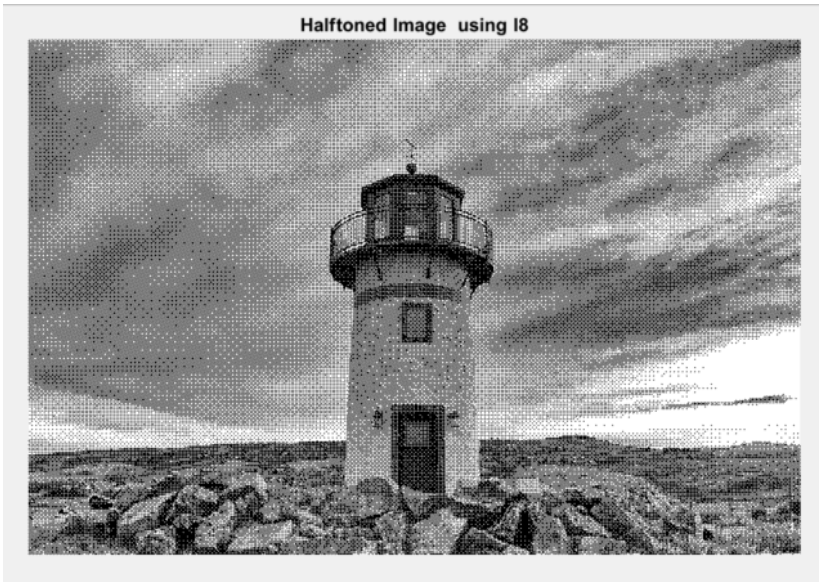


Picture 32: Random Thresholding Light House Image

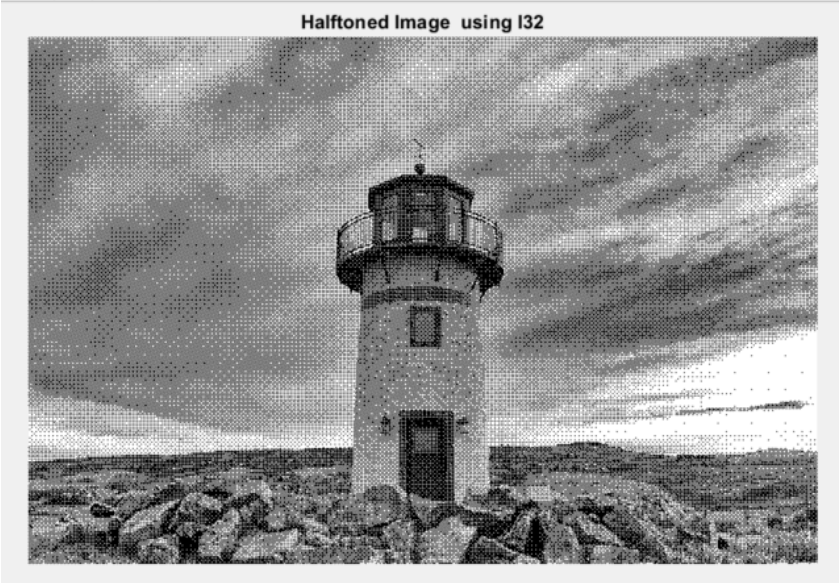
2. Dither Matrix



Picture 33: Dither Matrix I2



Picture 34: Dither Matrix I8



Picture 35: Dither Matrix I32

XX. Discussion:

The fixed threshold image yields a two-tone black and white representation, yet the quality of the lighthouse is not optimal; for more complex images, this kind of halftoning proves inadequate. With random thresholding, the image improves compared to fixed thresholding, but it suffers from significant graininess and noise.

In the dither matrix, all three matrices produce satisfactory results. Dither Matrix I2 creates a painting-like spread, while both I8 and I32 generate high-quality dither halftone

images. I prefer Dither Matrix method I8 due to its ability to maintain an image close to the original vision. In contrast, in I32, pixel separation becomes more noticeable.

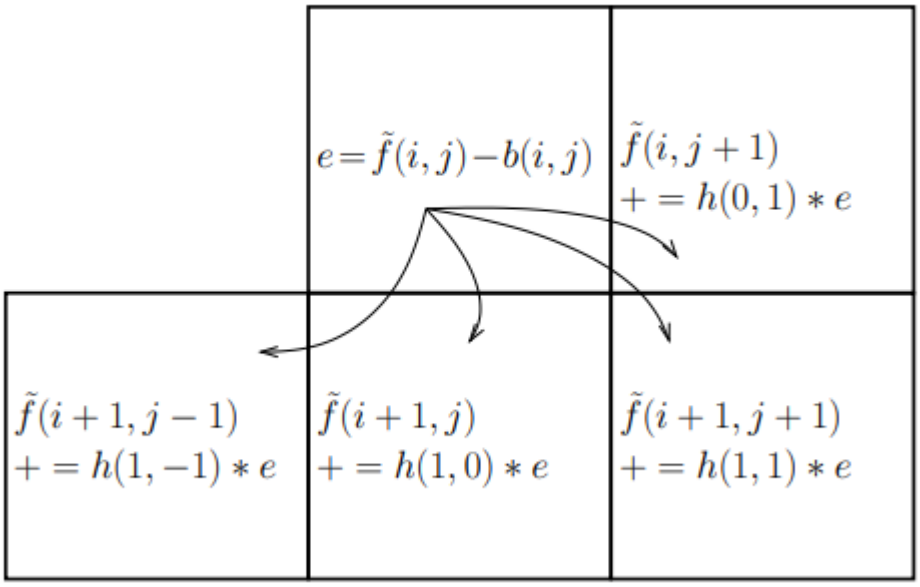
2b Error Diffusion:

XXI. Abstract and Motivation

Error diffusion is employed in image processing for several reasons. Unlike simple pointwise operations, error diffusion quantized each pixel based on its neighborhood, leading to smoother transitions and better preservation of image details. By traversing the image in raster order and distributing quantization errors forward, error diffusion mitigates artifacts and enhances overall image quality. This method can yield superior results, particularly in preserving fine textures and gradients, compared to traditional techniques. Its ability to maintain visual fidelity while reducing quantization errors makes error diffusion a preferred choice for high-quality image rendering in various applications, including printing, digital imaging, and multimedia processing.

XXII. Approach and Procedure

Pushing error forward in error diffusion is crucial for enhancing image quality. By distributing quantization errors to neighboring pixels, it minimizes artifacts and preserves details. This process ensures that errors are dispersed throughout the image, reducing noticeable discrepancies and producing smoother transitions between colors or intensity levels.



1. **Floyd and Steinberg:** This 1976 approach for error diffusion focuses mostly on diagonal error propagation by distributing quantization errors to surrounding pixels in a weighted way. Because of its ease of use and ability to reduce visual artifacts while maintaining image details, it is frequently utilized.

	■	7/16
3/16	5/16	1/16

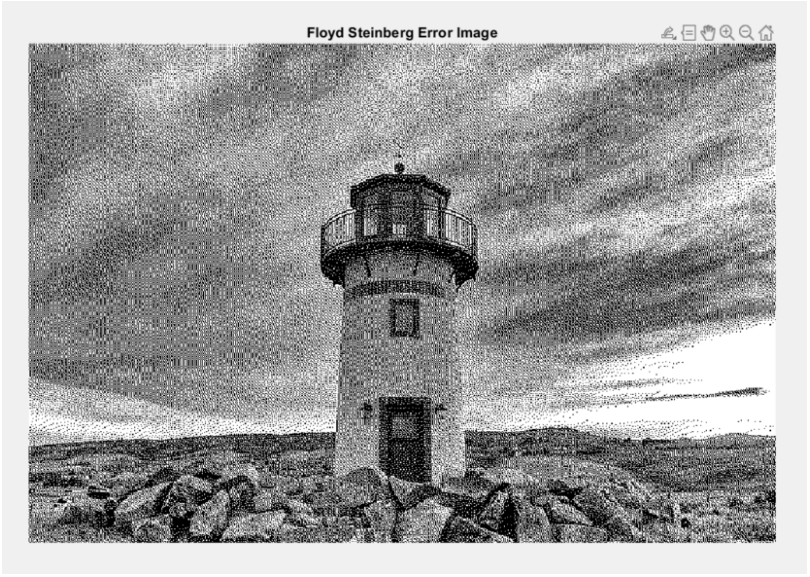
2. **Jarvis, Judice and Ninke:** This error diffusion technique, which was first presented in 1976, improves Floyd and Steinberg's algorithm by taking a wider pixel vicinity into account while propagating errors. It produces halftone images that are smoother and have fewer artifacts, which makes it appropriate for high-quality image reproduction.

		■	7/48	5/48
3/48	5/48	7/48	5/48	3/48
1/48	3/48	5/48	3/48	1/48

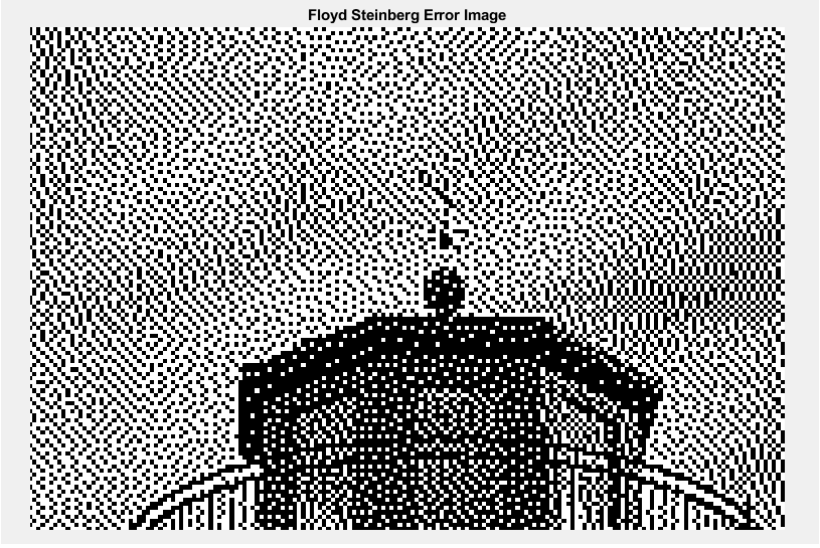
3. **Stucki**: By adding a more intricate error distribution pattern, the 1981 Stucki error diffusion algorithm improves upon existing error diffusion methods. It eliminates artifacts and improves image quality, especially in grayscale and color halftoning applications, by carefully diffusing mistakes to neighboring pixels.

$$\frac{1}{42}\begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 8 & 4 \\ 2 & 4 & 8 & 4 & 2 \\ 1 & 2 & 4 & 2 & 1 \end{bmatrix}$$

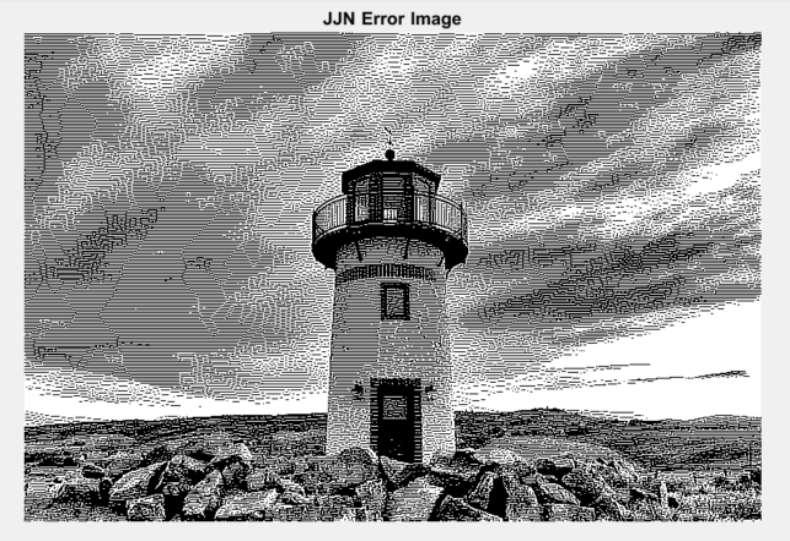
XXIII. Experimental Results:



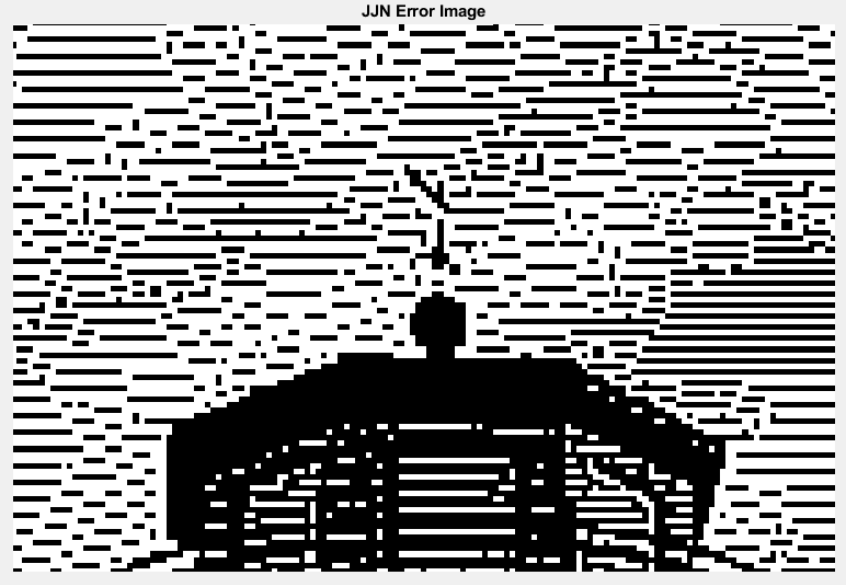
Picture 36: Floyd and Steinberg Error Diffusion Image



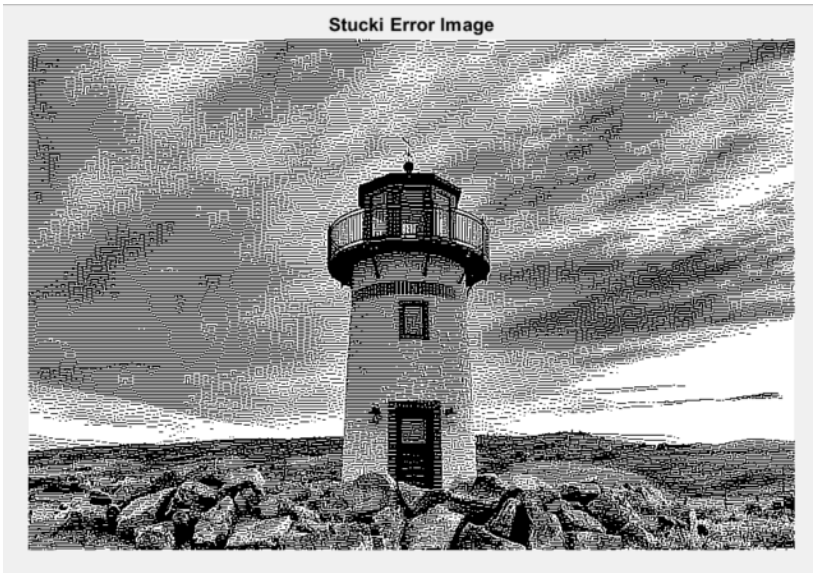
Picture 37: Floyd and Steinberg closeup



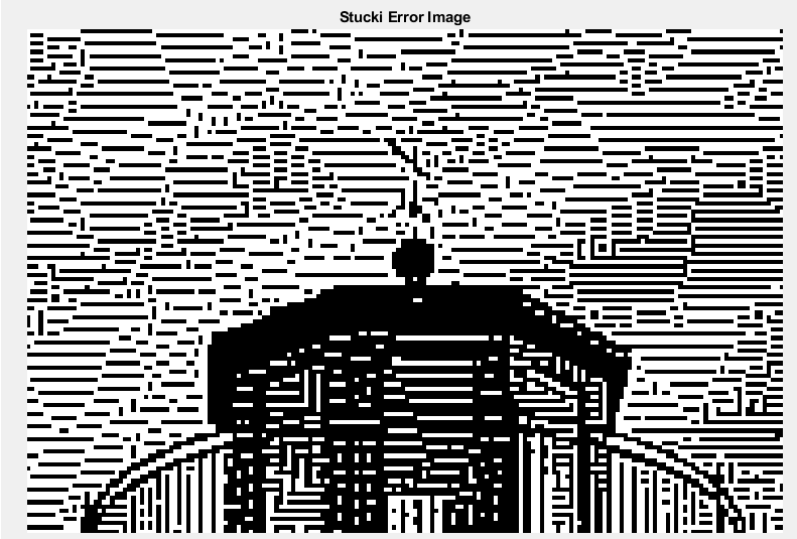
Picture 38: JJN Error Diffusion Image



Picture 39: JJN Error Diffusion closeup



Picture 40: Stucki Error Diffusion Image



Picture 41: Stucki Closeup

XXIV. Discussion:

When comparing the three error diffusion methods, it becomes evident that the Floyd-Steinberg (FS) diffusion image produces poorer output compared to the Jarvis, Judice, and Ninke (JJN) and Stucki images. Both JJN and Stucki images exhibit similar outputs, albeit with slight variations in error distribution patterns discernible upon closer inspection. Interestingly, the FS image closely resembles the I2 threshold matrix dither image. This observation suggests that while FS diffusion may not excel in certain aspects compared to JJN and Stucki methods, its output shares similarities with specific dithering techniques, implying nuances in their processing approaches.

After evaluating these halftoning techniques, the JJN error diffusion method and the I8 threshold matrix dithering image emerge as the most favorable options. Both techniques effectively mimic the original image, providing a close illusion of its details. Conversely, fixed thresholding proves to be the least useful method among these, offering limited fidelity and detail preservation in comparison.

If I were to implement the error diffusion algorithm, I would consider spreading the error in both upward and downward directions, instead of limiting it to one side only. By distributing the error across neighboring pixels in a balanced manner, the diffusion process could potentially yield more uniform and visually pleasing results. For instance, a distribution scheme could involve allocating error coefficients as follows:

4/16	4/16	1/16
2/16		7/16
3/16	5/16	1/16

3. Color Half-toning with Error Diffusion:

a.Separable Error Diffusion

XXV. Abstract and Motivation:

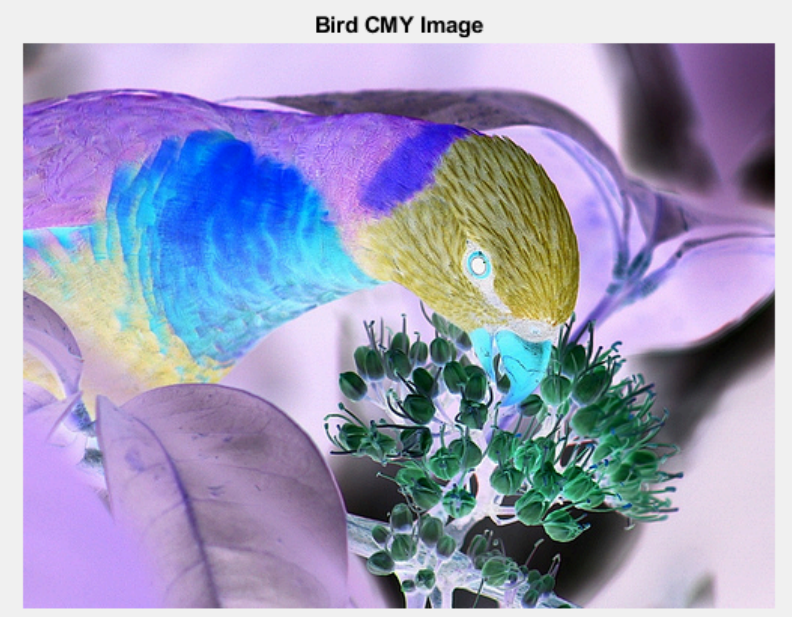
Separable error diffusion for color images involves treating each color channel independently during the error diffusion process. By separating the RGB channels, errors are diffused individually, maintaining color integrity

XXVI. Approach and Procedure:

First, we convert the image to CMY, and each layer is subsequently sent individually to the Floyd-Steinberg (FS) error diffusion algorithm. The resulting outputs are then concatenated, and the combined image is converted back to RGB format. This approach ensures that each color channel undergoes error diffusion independently, preserving color integrity throughout the process. By separating the CMY channels, artifacts are minimized, and image quality is enhanced. Finally, converting the concatenated image back to RGB format maintains compatibility while retaining the benefits of error diffusion for high-quality color image reproduction.

XXVII. Experimental Results:

Picture 42: Original Color Image



Picture 43: CMY Image

Picture 44: Separable Error Diffusion O/P

XXVIII. Discussion:

The output of separable error diffusion bears a striking resemblance to the FS error diffusion image observed earlier, albeit in color form. Despite the transition to color, the visual characteristics and quality closely mirror those of the grayscale FS method. This similarity underscores the effectiveness of the separable approach in maintaining consistency and fidelity across different color channels. The parallelism in results across both techniques reaffirms their reliability and robustness in preserving image details and reducing artifacts, ensuring high-quality color reproduction in error diffusion-based halftoning processes.

The only shortcoming is there is a potential for color misalignment, Independent error diffusion on separate color channels may result in color misalignment or inconsistencies, especially in regions with intricate color patterns or gradients.

b. MBVQ-based Error diffusion

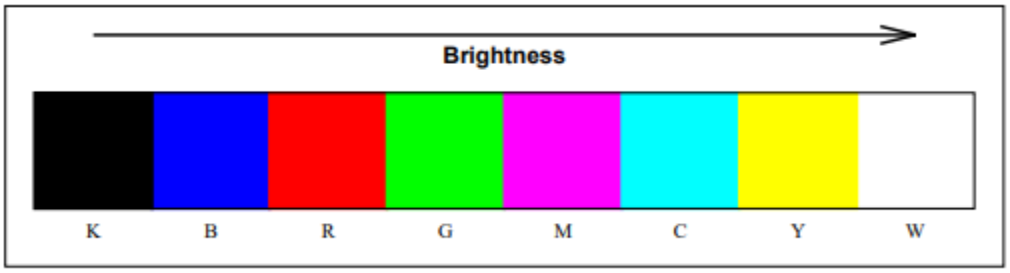
XXIX. Abstract and Motivation:

Color Diffusion is an innovative algorithm that incorporates the Minimal Brightness Variation Criterion (MBVC) to outperform separable Error Diffusion methods. Color Diffusion recognizes the natural brightness differences between colored dots, in contrast to traditional techniques that only apply monochrome halftoning concepts to color images, guaranteeing the best possible placement for each color component.

XXX. Approach and Procedure:

Key Ideas behind the MBVQ:

The fundamental concepts of the Minimal Brightness Variation Quantization (MBVQ) algorithm are based on a combination of color theory, human vision, and halftoning methods. The complexity of the human visual system (HVS), which interprets brightness fluctuations differently in different hues and viewing situations, is acknowledged. MBVQ maximizes halftone color selection to reduce artifacts and enhance visual fidelity by warping the color space to correspond with human perception.
The Minimal Brightness Variation Criterion, which states that halftone colors should reduce brightness fluctuations across the image to improve overall quality, is central to MBVQ. Ensuring conformity to the requirement and correct representation of intended colors, the algorithm meticulously selects participating halftone colors for every input color.
Important post-processing method Ink Relocation helps to maintain MBVC compliance while improving halftone patterns. Ultimately improving image quality and visual appeal, MBVQ synthesizes these ideas to create a strong foundation for color halftoning, giving perceptual integrity first priority and reducing brightness variances.

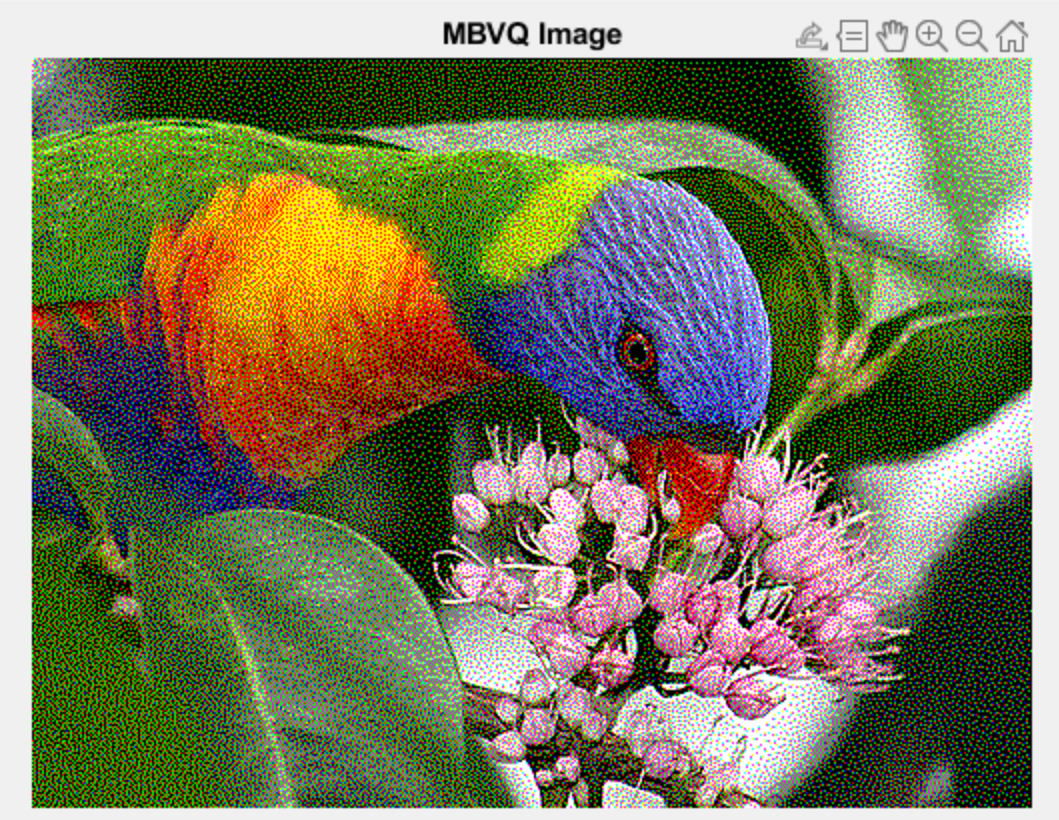


Algorithm:-

For each pixel (i, j) in the image do:

1. Determine MBVQ($RGB(i, j)$).
2. Find the vertex $v \in$ MBVQ which is closest to $RGB(i, j) + e(i, j)$.
3. Compute the quantization error $RGB(i, j) + e(i, j) - v$.
4. Distribute the error to "future" pixels.

XXXI. Experimental Results:



XXXII. Discussion:

Separable error diffusion bird image is darker in comparison with MBVQ bird image.Gap pixels in MBVQ typically take on more uniform tone colors, which improves brightness. On the other hand, gap pixels are black in separable error diffusion, reducing image brightness. This distinction emphasizes how MBVQ prioritizes color optimization and little brightness variance, producing images that are brighter and more visually appealing than those produced by separable error diffusion.