Cairo University
Faculty of Engineering
Credit Hours System

Task Canny, Hough & Harris

SBES160: Image Processing & Computer Vision

Group (9)

Group Members:

- Abdulmonem Elsherif
- Mohammed Tamer
- Ahmad Hussain Okasha
- Mohammed Osama

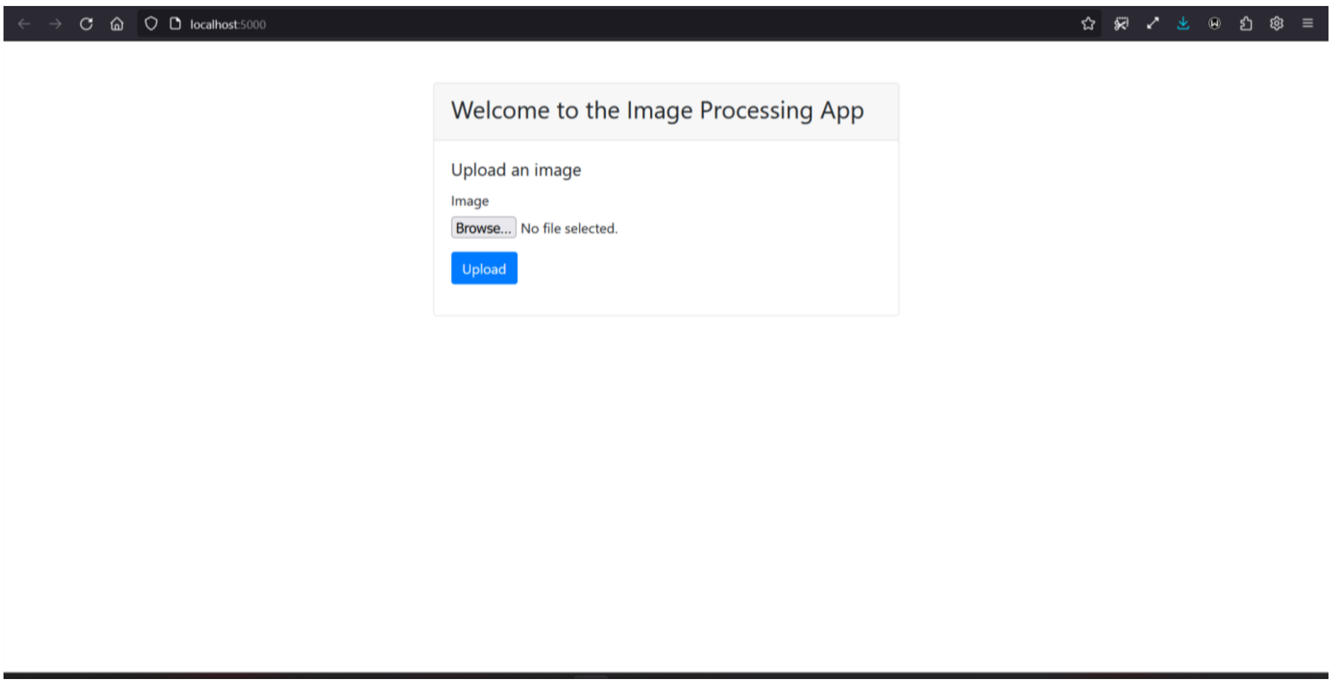
Table of contents

Introduction

This project is a web application that allows users to upload an image and apply various image processing operations on it. The operations include Canny edge detection, Hough line transform, and Harris corner detection.

Project Structure

Upon opening the web-app, the user is greeted with this screen:



Welcome to the Image Processing App

Upload an image

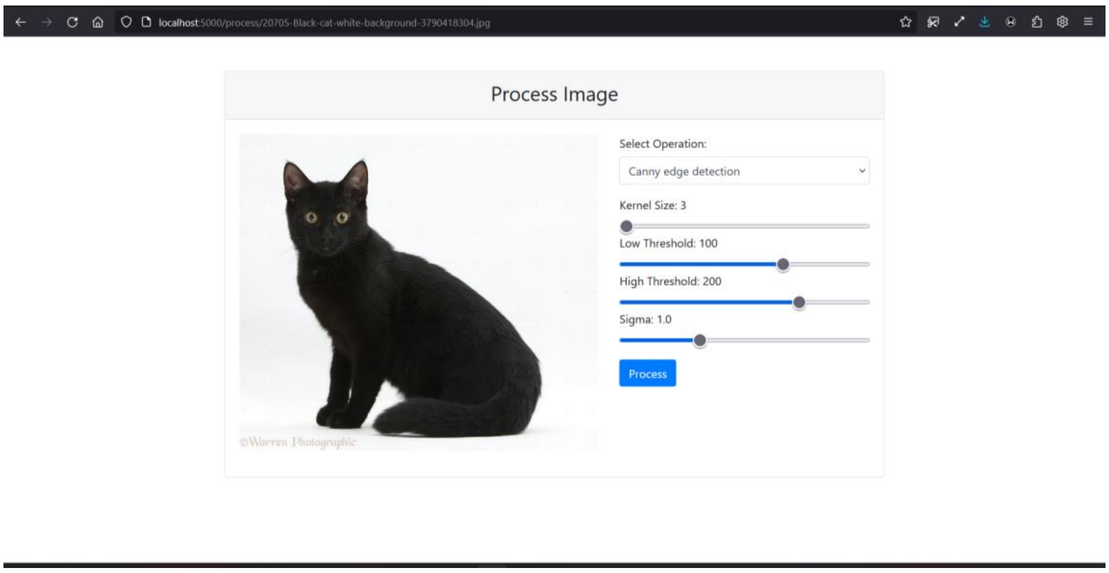
Image

No file selected.

1. **Form for Image Upload:** The form allows users to upload an image for processing. It includes a file input field for selecting an image file and a submit button styled with Bootstrap's primary color.
2. **Error Handling:** Error messages for the image upload are displayed in case of validation errors, such as selecting an unsupported file type.

3. **Styling:** The UI uses Bootstrap's CSS for styling, ensuring a responsive and visually appealing design.

When an image is chosen and uploaded said user would be redirected to another page where parameters could be set based on the desired operation:



localhost:5000/process/20705-Black-cat-white-background-3790418304.jpg

Process Image

Select Operation: Canny edge detection

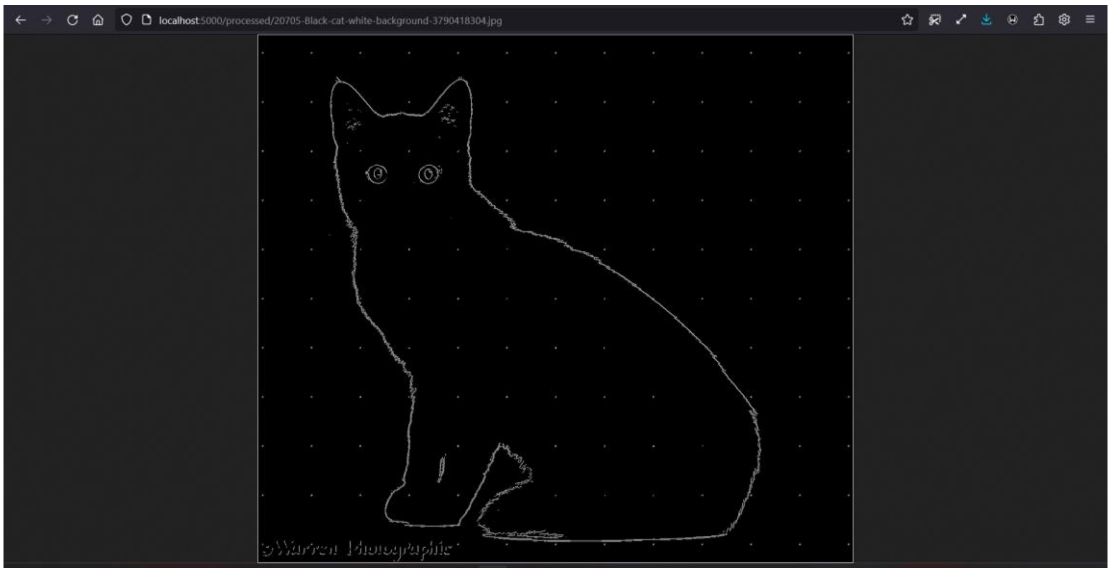
Kernel Size: 3

Low Threshold: 100

High Threshold: 200

Sigma: 1.0

output:



localhost:5000/processed/20705-Black-cat-white-background-3790418304.jpg

Image processing operations:

Canny Edge Detection

Canny edge detection is a popular image processing technique used to detect edges in images. It involves several steps to identify edges accurately while reducing noise. Canny edge detection is widely used in computer vision tasks such as object detection, image segmentation, and feature extraction.

Steps

1. **Converting to greyscale:** Only color intensity is relevant to this kind of filtration
2. **Gaussian filter:** applied to the image to reduce noise and unwanted details. This smoothing process helps in creating a more uniform gradient across the image, making it easier to identify edges. (removes unnecessary detail thus reducing false positives)
3. **Gradient Calculation:** After smoothing, the gradients of the image are calculated using techniques like Sobel operators. These gradients represent the rate of change of pixel intensities, highlighting areas with significant intensity changes, which often correspond to edges.
4. **Non-Maximum Suppression:** This step involves thinning the detected edges to a single-pixel width. It works by suppressing all gradient

values except for the local maxima along the edge directions. This ensures that only the sharpest edge points are preserved.

5. **Hysteresis Thresholding:** Finally, a hysteresis thresholding process is applied to classify edges as strong or weak. Pixels with gradient magnitudes above a high threshold are marked as strong edges, while those between high and low thresholds are considered weak edges. Weak edges are retained only if they are connected to strong edges, helping to eliminate noise and preserve continuous edges

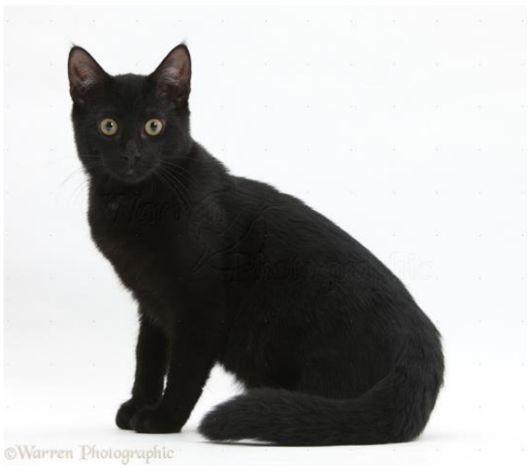
Variables

The changeable variables for Canny edge detection are defined through input range elements. Here's a breakdown of these variables:

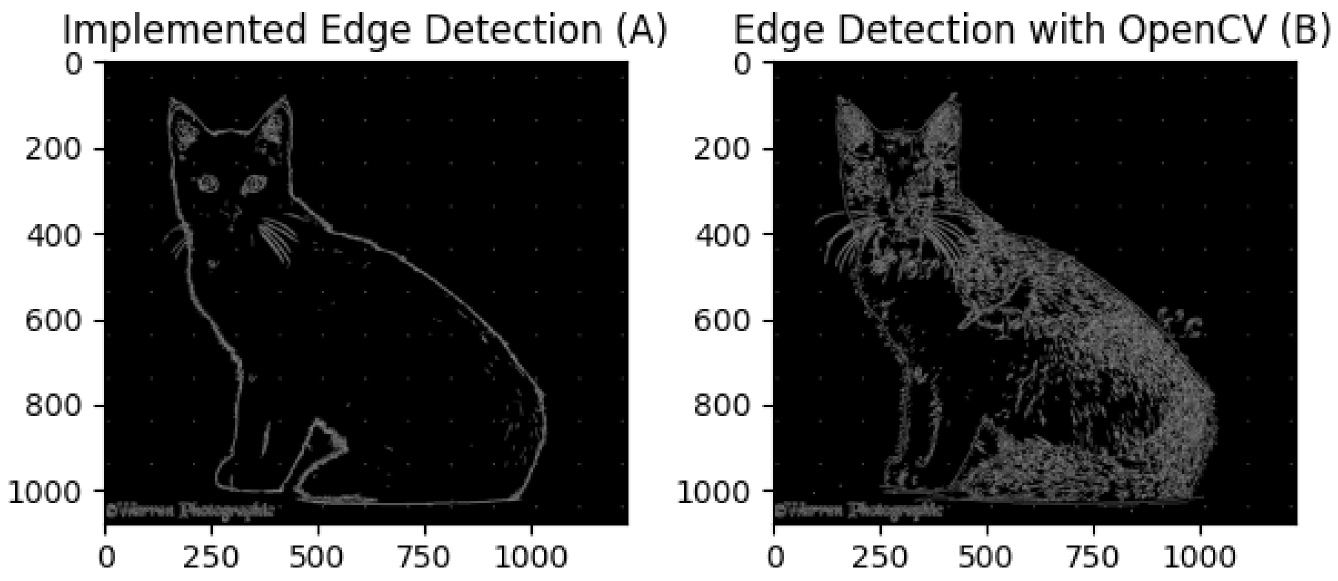
1. **Kernel Size (`canny_kernel_size`):**
   - Min Value: 3
   - Max Value: 7
   - Step: 1
   - Default Value: 3
   - This variable determines the size of the Sobel kernel used for edge detection. Increasing the kernel size can help capture larger edges but may also introduce more noise.
2. **Low Threshold (`canny_low_threshold`):**
   - Min Value: 1

  - Max Value: 150
  - Default Value: 100
  - This threshold value is used in the Canny algorithm to identify potential edges. Lower values result in more edges being detected, while higher values filter out weaker edges.
3. **High Threshold (`canny_high_threshold`):**
  - Min Value: 50
  - Max Value: 255
  - Default Value: 200
  - This threshold value is used in conjunction with the low threshold to determine which edges are strong (above the high threshold) and which are weak (between the low and high thresholds).
4. **Sigma (`canny_sigma`):**
  - Min Value: 0.1
  - Max Value: 3.0
  - Step: 0.1
  - Default Value: 1.0
  - Sigma is the standard deviation of the Gaussian filter applied to the image before edge detection. Higher sigma values result in more blurred images, affecting the edge detection process.

Results



©Warren Photographic



Implemented Edge Detection (A)
Edge Detection with OpenCV (B)
0
200
400
600
800
1000
0
250
500
750
1000
©Warren Photographic

Hough Transform

Steps

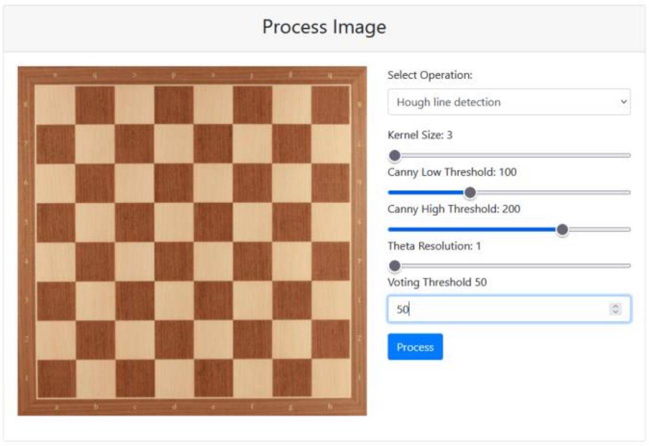
1. **Edge Detection**: The Hough transform typically starts with an edge detection algorithm like Canny to identify edges in the image. This step is crucial as the Hough transform works best with binary edge images where edges are represented as white pixels against a black background.
2. **Parameterization**: In this step, each edge point is parameterized to represent a line in a parameter space. For example, in the case of straight lines represented by the equation y=mx+cy=mx+c, each edge point corresponds to a line in the m−cm−c parameter space.
3. **Accumulation**: The parameterized lines from step two are then "voted" for in an accumulator array. Each edge point contributes to the accumulator cells that correspond to the parameter values of the line passing through that point. This accumulation process helps in finding the most prominent lines in the image.
4. **Peak Detection**: After accumulation, peaks in the accumulator array are detected, indicating the most significant lines in the image. These peaks correspond to the parameter values of lines that are most likely to represent actual features or structures in the image.
5. **Line Extraction**: The final step involves extracting lines based on the peaks detected in the accumulator array. These lines are reconstructed using the

parameter values corresponding to the peaks, providing a representation of the prominent straight lines present in the original image.

Variables:

1. **rho_resolution:** This variable determines the resolution of the distance parameter (ρ ρ) in the Hough space. It specifies how finely the distance values are sampled along the rho axis. A smaller rho resolution leads to a more detailed Hough space but requires more computational resources.
2. **theta_resolution:** The theta resolution variable sets the granularity of the angle parameter (θ θ) in the Hough space. It controls the step size between different angle values considered during the Hough transform process. A smaller theta resolution provides more precise angle detection but may increase processing time.
3. **threshold:** The threshold parameter sets the minimum number of votes required for a line to be considered a valid detection in the Hough space. Increasing the threshold value can filter out weaker or less prominent lines, resulting in a more robust line detection process. However, setting it too high may lead to missing valid lines.

Outputs and realizations:


Process Image

Select Operation: Hough line detection

Kernel Size: 3

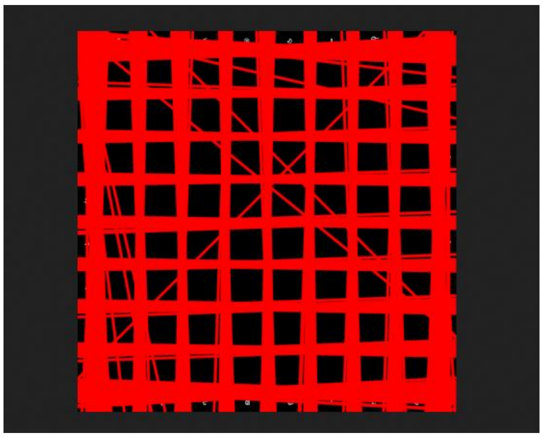
Canny Low Threshold: 100

Canny High Threshold: 200

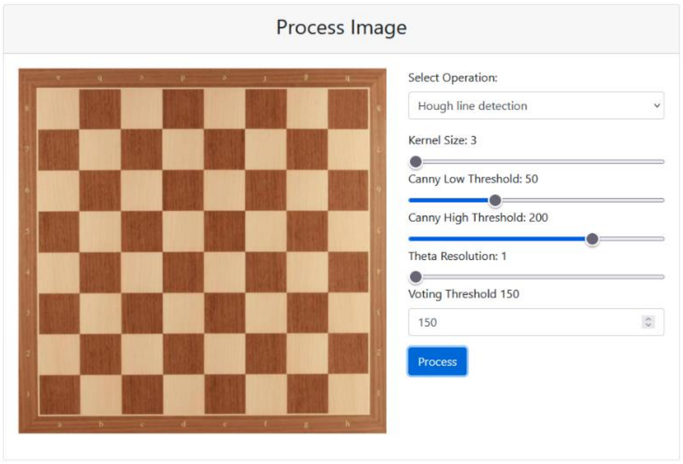
Theta Resolution: 1

Voting Threshold 50

50



As we can see here, putting a low value of threshold and θ resolution with 1 is going to detect a lot of lines because 50 votes in the accumulator is a low value as there are a lot of edges in this image so the output will not be accurate. That is why we need to increase the threshold value to something between 150-200 for accurate line detection.


Process Image

Select Operation: Hough line detection

Kernel Size: 3

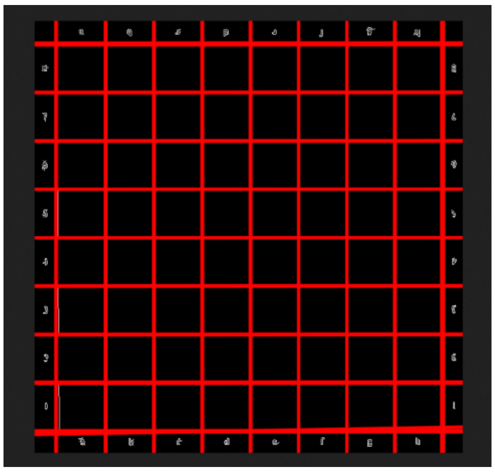
Canny Low Threshold: 50

Canny High Threshold: 200

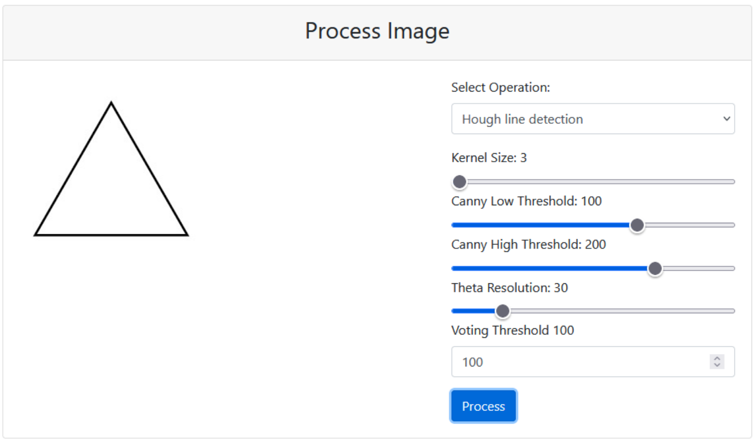
Theta Resolution: 1

Voting Threshold 150

150



When we changed the threshold to 150. The output became much better as now there must be at least 150 votes in the accumulator array for the line to be accepted and plotted.


Process Image

Select Operation: Hough line detection

Kernel Size: 3

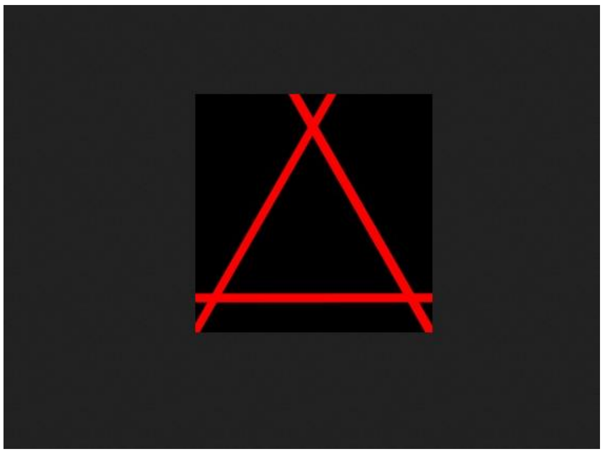
Canny Low Threshold: 100

Canny High Threshold: 200

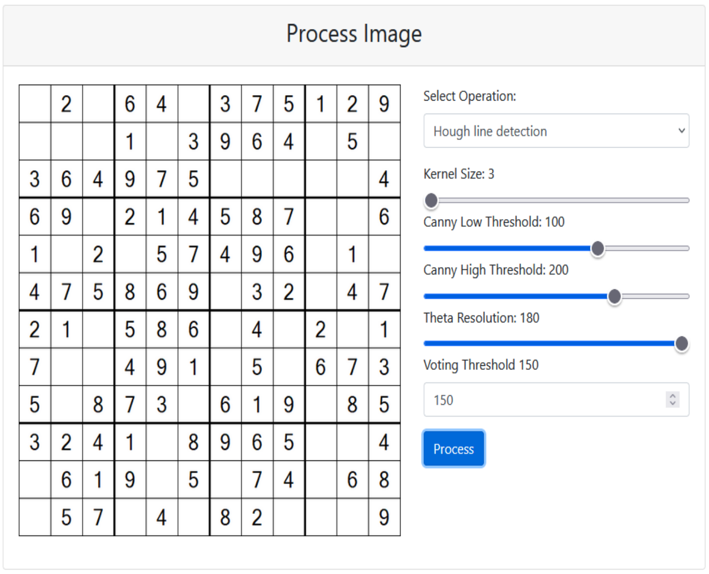
Theta Resolution: 30

Voting Threshold 100

100



In this triangle image, changing the θ resolution to 30 degrees yields a correct output. As the change in detection of θ will be every 30 degrees so it is going to yield a correct line detection to all the sides of the triangle as the change of edge detection will be equal to the exterior angle of the triangle which is correct if consider that θ starts from the vertical y-axis and moves counter-clockwise


Process Image

	2		6	4		3	7	5	1	2	9
			1		3	9	6	4		5	
3	6	4	9	7	5						4
6	9		2	1	4	5	8	7			6
1		2		5	7	4	9	6		1	
4	7	5	8	6	9		3	2		4	7
2	1		5	8	6		4		2		1
7			4	9	1		5		6	7	3
5		8	7	3		6	1	9		8	5
3	2	4	1		8	9	6	5			4
	6	1	9		5		7	4		6	8
	5	7		4		8	2				9

Select Operation: Hough line detection

Kernel Size: 3

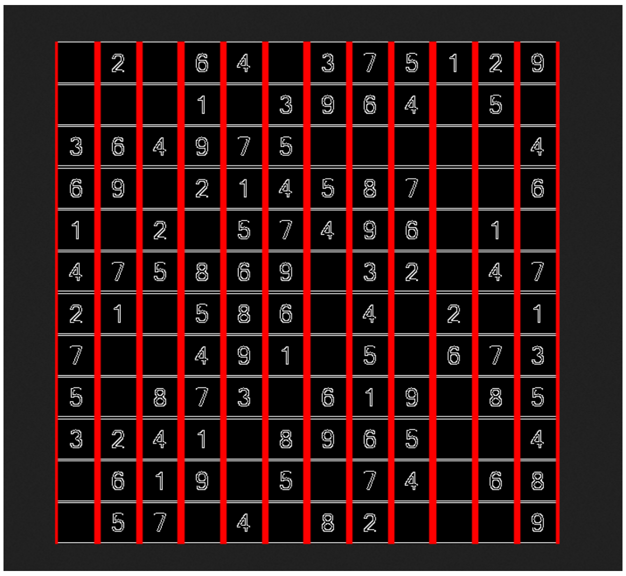
Canny Low Threshold: 100

Canny High Threshold: 200

Theta Resolution: 180

Voting Threshold 150

150



In this case, setting the θ resolution to 180 yields only vertical lines which makes sense because it is going to check for edges in only one way which is vertical. (Accumulator array will only have one column for thetas)

Harris Corner Detection

Steps

1. **Gradient Calculation**: Harris corner detection starts with calculating the gradients of the image using techniques like Sobel operators. These gradients provide information about the intensity changes in different directions across the image.
2. **Structure Tensor Calculation**: The structure tensor is computed based on the gradients calculated in the previous step. It consists of elements that represent the local image structure and orientation. The elements of the structure tensor are derived from the products of gradients in x and y directions.
3. **Corner Response Function**: Using the elements of the structure tensor, a corner response function is computed at each pixel location. This function evaluates the likelihood of a pixel being located at a corner based on the eigenvalues of the structure tensor. Higher eigenvalues indicate strong corner-like structures.

4. **Non-Maximum Suppression**: After computing the corner response function, non-maximum suppression is applied to identify local maxima in the response function. This step helps in selecting the most significant corner points while suppressing non-maximum responses.
5. **Thresholding and Corner Selection**: Finally, a thresholding mechanism is applied to the corner response values to select corners above a certain threshold. These selected corners represent the detected corner points in the image, which are often key points used in feature matching and object recognition tasks.

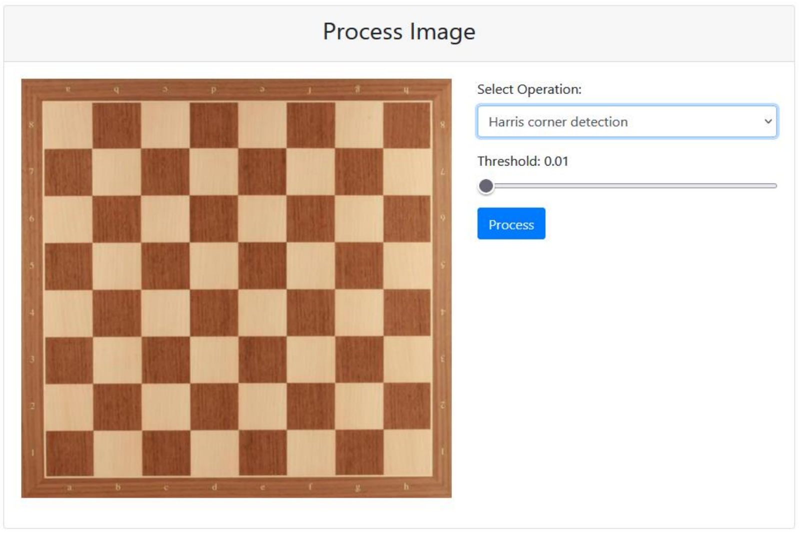
Variables

1. **Window_size:**
    - Default Value: 3
    - Description: This variable determines the size of the window used for computing the structure tensor and applying Gaussian smoothing. A larger window size can capture more global image features but may also increase computational complexity and blur small-scale details.
2. **k:**
    - Default Value: 0.04
    - Description: The kk parameter in the Harris corner response formula R=det-k×trace2R=det-k×trace2 controls the sensitivity of the corner detector. Increasing kk emphasizes corners with stronger intensity

gradients but may also lead to more false positives or miss weak corners.

3. **Threshold:**
   - Default Value: 0.01
   - Description: This threshold value is used in the non-maximum suppression step to filter out weak corner responses. Increasing the threshold can result in fewer detected corners, while decreasing it may detect more corners, including weaker ones.

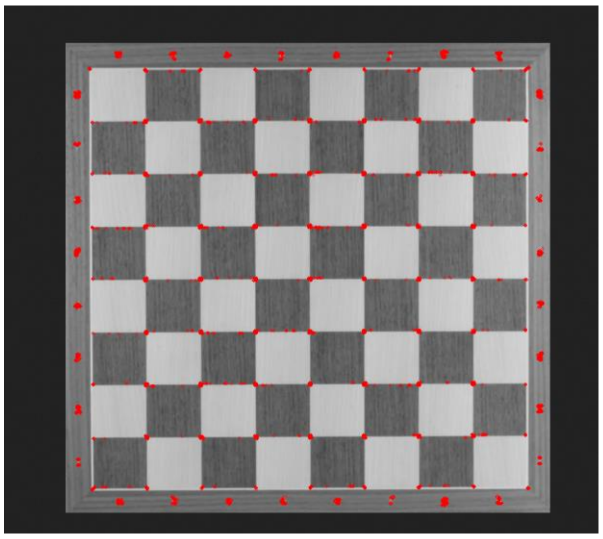
Outputs and realizations:



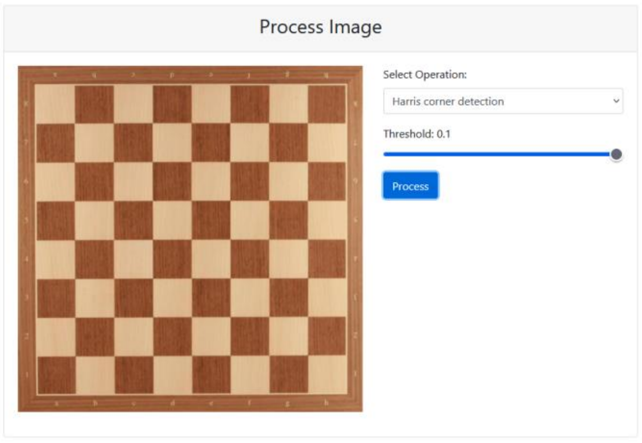
Process Image

Select Operation: Harris corner detection

Threshold: 0.01



The threshold values in harris means a specific value of R that if it is surpassed, the specific pixel is going to be considered a corner according to the values of lamda 1 and lamda 2. That is why in the example above. Decreasing the threshold is going to yield a lot of corners, even it is weak or irrelevant corners as we can see in the numbers and letters on the side of the chess board.

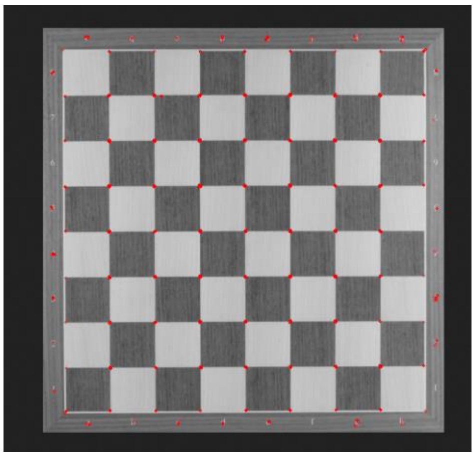


Process Image

Select Operation:

Harris corner detection

Threshold: 0.1



On the contrary, when we increase the value of the threshold. The edge detection becomes much better and the weak and irrelevant edges are filtered from the picture.