Road and Field Boundary Detection in Satellite Imagery

Group 15

Harshavardhan Reddy Mallannagari
Student ID – 11603389
harshavardhanreddymallannagari@my.unt.edu

Preethi Vahitha Sankineni
Student ID -11607639
preethivahithasankineni@my.unt.edu

Sasidhar Yalamanchili
Student ID – 11573866
sasidharyalamanchili@my.unt.edu

University of North Texas

PROBLEM STATEMENT

Road and field boundary detection in satellite or aerial view images is very important in specific areas such as agriculture and urban plan designing. To perform tasks such as managing the field areas supplying resources, it is necessary that the maps of the roads and fields should be accurate and detailed. The clarity of the images is the key. But in this case, the landscapes are very complex and complicated with many interconnections. Hence, the automation process will be a little bit difficult. To overcome this, the project requires a method that can identify the roads and field boundaries in increased pixels to have large and clear images.

Thus, the main problem statement of this project is to detect the roads and field boundaries. And to differentiate the roads and field boundaries from other line features such as waterbodies, railway lines, etc. To achieve this we use different feature engineering methods such as Canny edge detection, Sobel edge detection, Harris Corner, Hough Transform, and DoG (Difference of Gaussian) filter, etc.

DATA

The project utilizes two data sets. The images utilized for this project are satellite-captured aerial images of various places in the United States. These images provide visual information about the landscape. The dataset used in this project is given by Professor Xiaohui Yuan.

The first set is the development data, which is used to train the models, and the second set is for evaluation and validation, which determines how good and efficient the model is. The images in the dataset are provided by the U.S. Department of Agriculture from the agricultural fields in the USA. They are in jpeg format. These images are cropped from the sections of the large images, and their pixel size is 1 m^2.

So, we have a total of 9 satellite images of the land that consists of fields, mountains, baren land, and lakes etc. From the given images the main objective of the project is to identify the road and the field boundaries. The below image fig 1 is one of the images in the given dataset.

fig.1. field.jpg

DESCRIPTION

The main idea is to create and develop a system that is used to identify the fields and road boundaries in satellite images. For the given project to be successful we need to perform various image processing methods, edge detection techniques and accuracy evaluations.

So, for the preprocessing part of the project, we first collect the data and apply some filters that help in enhancement of the images. The ground truth dataset of the images is created which is used to find the accuracy and evaluate other performance metrics of the selected working models. Then we use the Canny and Sobel edge detection methods to detect the edges in the images and apply Harris Corner algorithm to find points in the image. So, the points detected by Harris Corner will later help in identifying the boundaries. Hough Transform is applied to detect the straight lines in the images that are equivalent to the boundaries of the fields and roads. While applying each method we also try to check the accuracy of the output obtained by the model with the corresponding ground truth generated.

METHOD

A detailed explanation of the methodology applied is discussed here.

Data Acquisition and preprocessing: This includes techniques such as Noise reduction, increasing the contrast, resizing the image, using filters for better results, etc.

- **Gaussian Filter**: This is an image preprocessing technique. To prevent noise and to get smooth images, Gaussian Filtering method is used.
- **Median Filter**: Sorts the pixels in increasing order and then takes the median value. This filter is applied to avoid blurring edges and details in an image.
- **Histogram Stretching**: Histogram Stretching is an image analysis method. It is an image enhancement technique which gives clear images with enhanced contrast and clarity. The histogram obtained is expected to have a full range of colors.
- **Histogram Equalization**: This technique performs uniform distribution on the image histogram in order to obtain high contrast. Also, the image intensity will be adjusted resulting in vivid images.

- **DoG Filter**: It is the difference computed between two gaussian blurred images with different thresholds. It provides object boundaries at various scales.
- **Hough Transform**: Hough Transform is a feature engineering method which detects objects that can be expressed in mathematical equations. This technique is used to map an edge image. It can identify the lines or curves present in the images.

1. **Ground Truth of the images:** Ground truth of the images in the dataset acts as an asset in evaluating the models' predictions. We compare the results obtained from the models with the ground truth images. They serve as a benchmark for the models' performances. To get the ground truth images, we perform manual annotation. Arivis Cloud website or apeer.com was used to do the manual annotation [1].

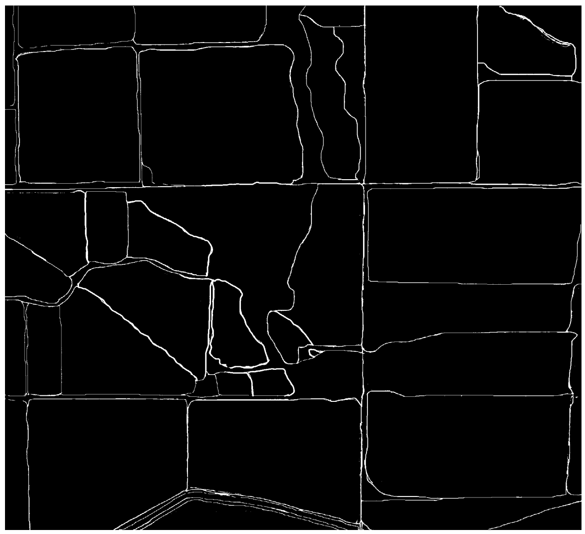

Fig.2. Ground Truth of field.jpg

2. **Model Design:** The model is designed using MATLAB including the steps mentioned below:

Pre-Analysis:

- Only one image (field.jpg) is used for the pre analysis. The manual annotation to get the ground truth image is also done for all the images in the dataset.
- To perform the pre-analysis, first we read the image using imread() and the groundtruth image that we annotated manually. We also tried to obtain some information about the image like bit depth and its HSV components. Later we converted the image into grayscale image.
- To confirm that the image is converted into grayscale image, we check the dimensions of the image. If the image has a single channel, then it means that it is converted into grayscale. The dimensions of the image are 2048 pixels (height) x 2048 pixels (width) x 1 channels.
- The dimensions of all the ground truth images are identified in the sizeOfImages.mlx file. The output contains the number of rows, columns, and channels of each image. In this code we are not processing or analyzing the images but just trying to understand their structure.

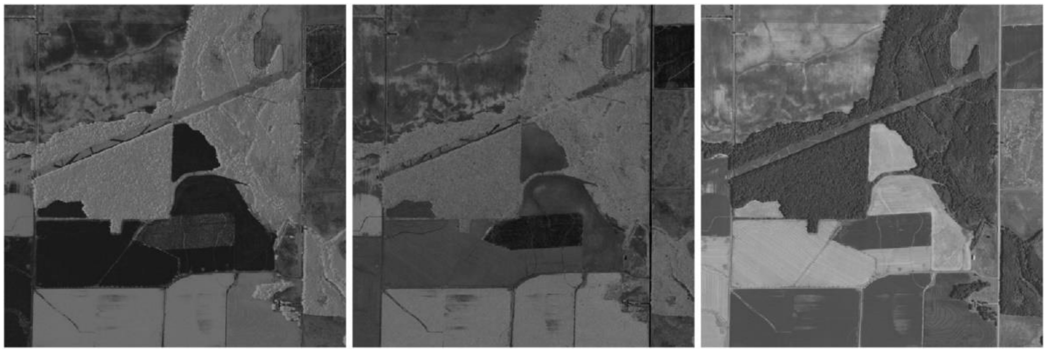

Fig.3.HSV Format of L97b

```
        2048        2048           3

pathGT = "C:\Users\harsh\OneDrive\Desktop\UNT Subject
disp(size(imread(pathGT)));

        2048        2048           3

pathGT = "C:\Users\harsh\OneDrive\Desktop\UNT Subject
disp(size(imread(pathGT)));

        2048        2048           3

pathGT = "C:\Users\harsh\OneDrive\Desktop\UNT Subject
disp(size(imread(pathGT)));

        2048        2048           3
```

Fig.3.1 Size of images

- Let's convert the grayscale image into a binary image to view the boundaries clearly. The binary image obtained is shown below:

Fig.3.2. Binary image of field.jpg

The workflow for the two main methods is shown below:

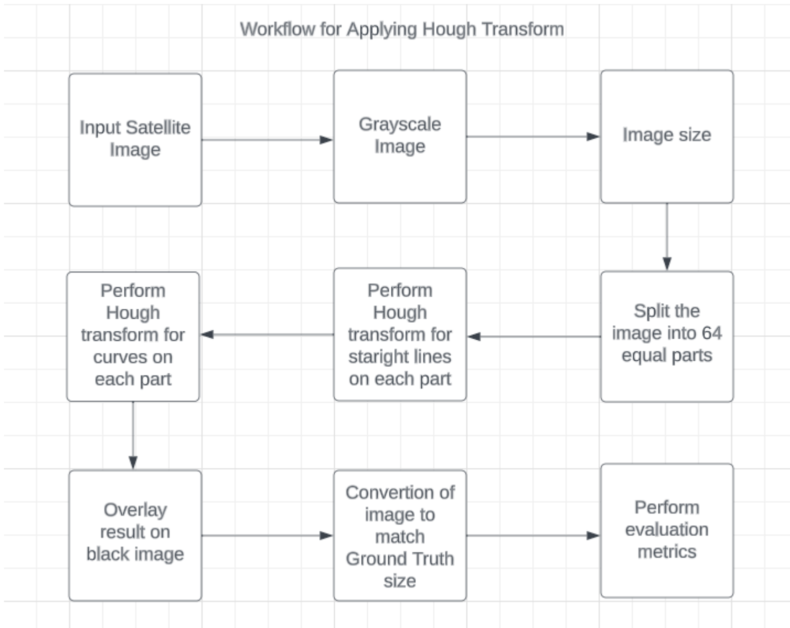

Fig.4. Workflow for Applying Hough Transform

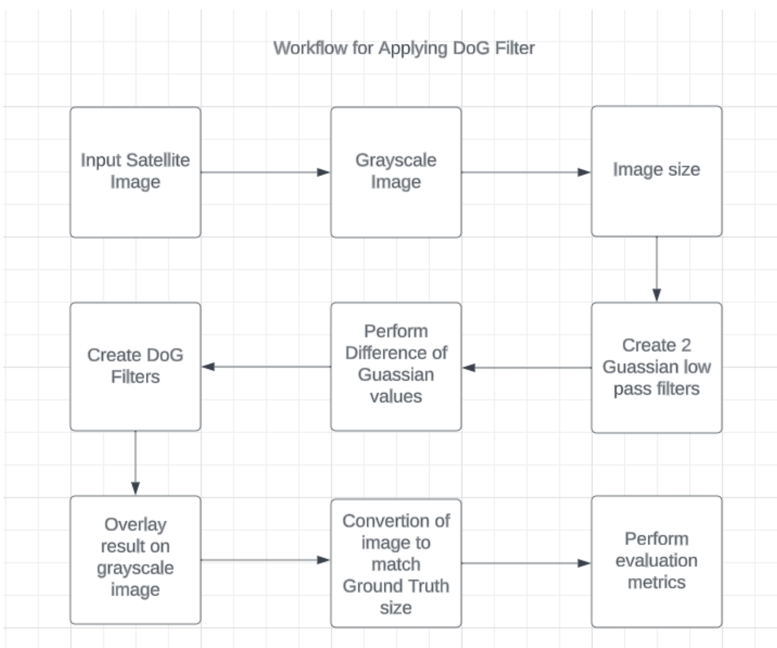

Fig.4.1. Workflow for Applying DoG Filter

Applying different types of Edge Detectors –

Sobel Edge Detector – This filter is a built–in function in MATLAB. It is a technique to detect edges in an image both vertically and horizontally. Initially, this method is implemented without using any filters on the image data. Later, Sobel Edge Detector method is applied under various threshold values to identify the best threshold value that provides the highest accuracy. We use ground truth as a benchmark to find out the accuracy of the model. The intersection and union values of the ground truth image and the initial image are calculated to find the accuracy. Filters are added to the grayscale image of the original image such as Gaussian filter, Histogram Equalization, Histogram Stretching and Median filter. The accuracy and threshold values of the Sobel Edge Detector after applying each filter are observed.

So, the code reads the original image and then its corresponding ground truth image. At first, the RGB image is converted into a grayscale image using rgb2gray() method. Then the Sobel Edge Detector with a specific threshold value is applied to the grayscale image and the IoU accuracy is calculated between the edges that are detected using Sobel Edge Detector function and the ground truth image which has been manually annotated. Then in the code we tried to compare the current accuracy and the maximum accuracy obtained so far. If the current accuracy is higher than the max accuracy, we update it and note the corresponding threshold during this optimization process.

```
Threshold : 0.070000        Accuracy : 6.978179
Threshold : 0.080000        Accuracy : 6.081869
Threshold : 0.090000        Accuracy : 5.208193
Threshold : 0.100000        Accuracy : 4.322272
Threshold : 0.110000        Accuracy : 3.453242
Threshold : 0.120000        Accuracy : 2.743991
Threshold : 0.130000        Accuracy : 2.163050
Threshold : 0.140000        Accuracy : 1.674081
Threshold : 0.150000        Accuracy : 1.244510
Threshold : 0.160000        Accuracy : 0.885450
Threshold : 0.170000        Accuracy : 0.584040
Threshold : 0.180000        Accuracy : 0.373930

Max Threshold : 0.050000        Max Accuracy : 7.849831
```

Fig.5. Sobel Edge max Threshold & max Accuracy

Gaussian filtering with different variance, median filtering with 3x3 and 5x5 filters, histogram stretching, and equalization are applied to the gray image followed by the Sobel edge detection one after the other. The following table shows how the accuracy varies for each filter.

Model Name	**Filter**	**Threshold**	**IOU**
Sobel	None	Default	7.875558
Sobel	Gaussian	0.05	7.849831
Sobel	Median	0.01	6.200332
Sobel	Histogram Equalization	Default	6.428086
Sobel	Histogram Stretching	Default	6.417841

Table 1. Sobel Edge Detector Accuracy

The image below shows output for the Sobel edge detector which has been tested with different thresholds:

Fig.6. Sobel Edge Detector for field.jpg

Canny Edge Detector – This filter aims at finding the strongest edges and the edges that are connected. This technique blurs the image to suppress noise and detects the edges accurately with very few false edges. Canny edge detector is implemented without using any filters on the image data. Later, various threshold values are used as parameters to identify the best threshold value that provides the highest accuracy. We use ground truth as a benchmark to find out the accuracy of the model. The intersection and union values of the ground truth image and the initial image are calculated to find the accuracy. The following filters are added to the grayscale image of the original image - Gaussian filter, Histogram Equalization, Histogram Stretching and Median filter. And the accuracy and threshold values of the Canny Edge Detector are calculated when each of the filters is applied to the image. The accuracy of Canny Edge Detection was evaluated using the fingIOUAccuracy() method i.e., Intersection over Union(IoU) metric is 4.618289.

Here's a detailed explanation of the Canny Edge Detector Model. In the beginning, the code reads two images, the original image field.jpg and its ground truth image which is manually annotated. After converting the rgb image into gray image, different variables such as maxAcc to get the maximum accuracy, threshold1 and threshold2 where t1 and t2 are given as the starting values for the thresholds. Inside the loop, we apply the Canny Edge detector to the gray image and calculate the accuracy using the IoU metric for the result obtained from canny edge function and the ground truth image. Among the thresholds and the accuracy obtained for each iteration, the maximum accuracy and its corresponding maximum threshold is displayed. The output for the same is shown below.

```
Threshold1 : 0.820000       Threshold2  : 0.910000     Accuracy : 0.000000
Threshold1 : 0.830000       Threshold2  : 0.920000     Accuracy : 0.000000
Threshold1 : 0.840000       Threshold2  : 0.930000     Accuracy : 0.000000
Threshold1 : 0.850000       Threshold2  : 0.940000     Accuracy : 0.000000
Threshold1 : 0.860000       Threshold2  : 0.950000     Accuracy : 0.000000
Threshold1 : 0.870000       Threshold2  : 0.960000     Accuracy : 0.000000
Threshold1 : 0.880000       Threshold2  : 0.970000     Accuracy : 0.000000
Threshold1 : 0.890000       Threshold2  : 0.980000     Accuracy : 0.000000
Threshold1 : 0.900000       Threshold2  : 0.990000     Accuracy : 0.000000

fprintf(" Max Threshold1 : %f    Threshold2 : %f      Max Accuracy : %f

 Max Threshold1 : 0.150000     Threshold2 : 0.240000      Max Accuracy : 8.080580
```

Fig.7. Canny Edge max Threshold and max Accuracy

Gaussian filtering with different variance, median filtering with 3x3 and 5x5 filters, histogram stretching, and equalization are applied to the gray image followed by the Canny edge detection one after the other. The following table shows the variance in the accuracy for each filter.

Model Name	**Filter**	**Threshold**	**IOU**
Canny	None	Default	4.363802
Canny	Gaussian	0.15 – 0.24	8.080580
Canny	Median	0.15 – 0.24	7.679530
Canny	Histogram Equalization	Default	3.991448
Canny	Histogram Stretching	Default	6.928220

Table 2. Canny Edge Detector Accuracy

The output for canny edge detector for the image field.jpg is shown below:

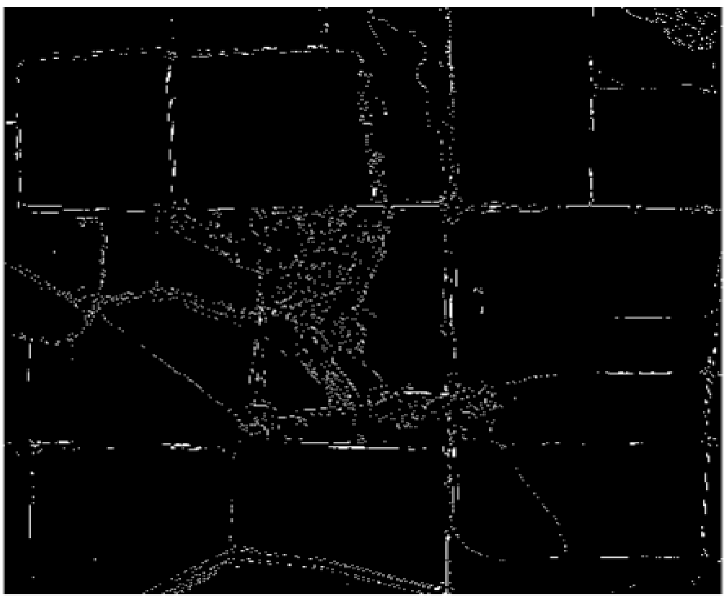

Fig.8 Canny Edge Detector for field.jpg

Haris Corner Detection –

Harris Corner Detector is used to identify a corner by looking through a small window. The window will be shifted in different directions to compute the intensity variations. The Harris Corner detector is invariant to the image rotation. So, in the beginning the code converts the rgb image into grayscale image. Later, we define the parameters of the Harris Corner detector such as sigma (Standard deviation for Gaussian Smoothing), window_size (Size of the window for the local neighborhood) and k (Harris corner constant). Since Harris Corner detection is a multi-step process, we first compute the image gradients using Sobel operators in both x and y dimensions. Additionally, we derive the elements of the Harris matrix and apply gaussian filter for smoothing. A threshold value is applied to the Harris corner response that is computed with the help of the Harris matrix elements. The corner points are detected using this threshold value and which are then highlighted as shown in the image below.

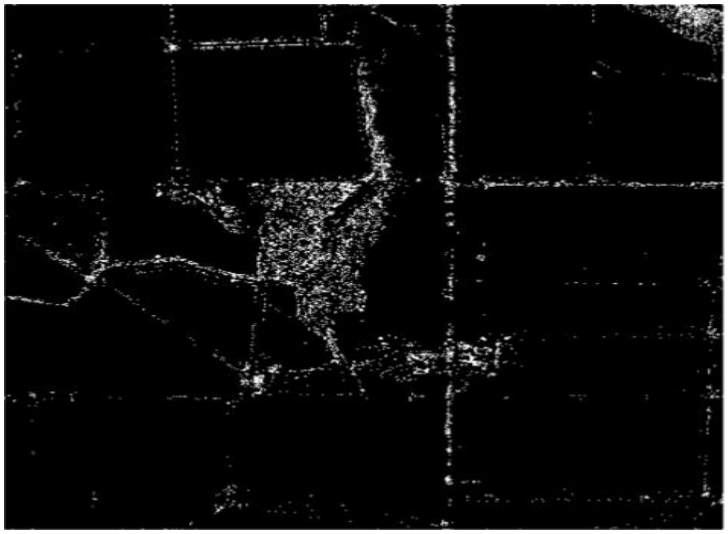

Fig.9 Harris Corner Detection on field.jpg

In order to evaluate this method, we calculated the accuracy using the Structural Similarity Index (SSIM) which is used to measure the similarity between images. The result obtained is nearly 49.04 for the Harris Corner Detector.

Hough Transform –

Hough Transform is a feature engineering method which detects objects that can be expressed in mathematical equations. This technique is used to map an edge image. It can identify the lines or curves present in the images. Hough transform works well if the input image is divided into sub-parts. When we perform Hough transform directly on the input image, we get virtual lines as shown in the below image. This is not an accurate way of working with Hough Transform. Hence, we split the image into 8x8 grid before performing Hough transform.

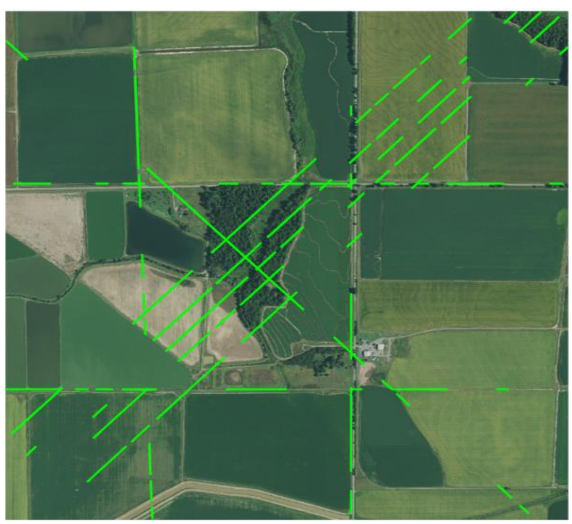

Fig.10. Virtual Lines in Hough Transform on field.jpg

In this project we performed Hough Transform in two ways. At first, we only considered the image divided into an 8x8 grid and later we even considered the curves in the image. So, the outputs and the accuracy are derived and calculated separately for each way. So, the first part works by initializing an empty image of the same size such as the input image and then the input image is converted into gray image while iterating on the sub-parts of the 8x8 grid. Later, Hough transform is performed on each part of the 8x8 grid on which edges are detected using the Canny edge detection algorithm. These detected lines are drawn on the blank image in the next step of the process. The final image contains the empty image with black background on which the lines are drawn corresponding to the edges detected on each sub-part of the input image. The output obtained by performing Hough transform on the L88a.jpg image is shown below.

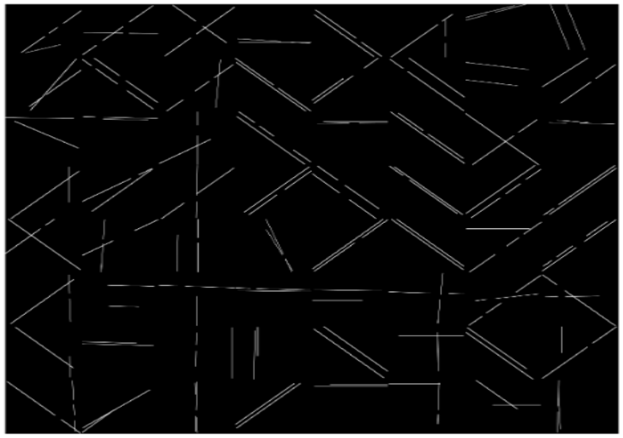

Fig.11. Hough Transform on L88a.jpg (lines)

In addition, using Hough Transform we tried to identify the curves present in the input image along with the edge's detection. This is the extension of the previous code. The process of how Hough Transform is performed on the sub-images of an input image using both polynomial curves and detected edges is detailly explained below. We first read the input image and then divide it into 8x8 sub-parts after identifying their dimensions. After we get the sub-parts we calculate their dimensions i.e., number of rows and number of columns. Like in the earlier time, we create a blank image with black background. Now we will loop over each part of the input image and convert each part into grayscale. Later, we make use of the Canny Edge detector to discover the edges in each sub image. These detected lines are drawn on the image with black background. Then the curve fitting is done with the help of a second- degree polynomial function polyfit(). The discovered curves within each sub-image are drawn on the image. This completes the visualization of lines and curves using Hough Transform.

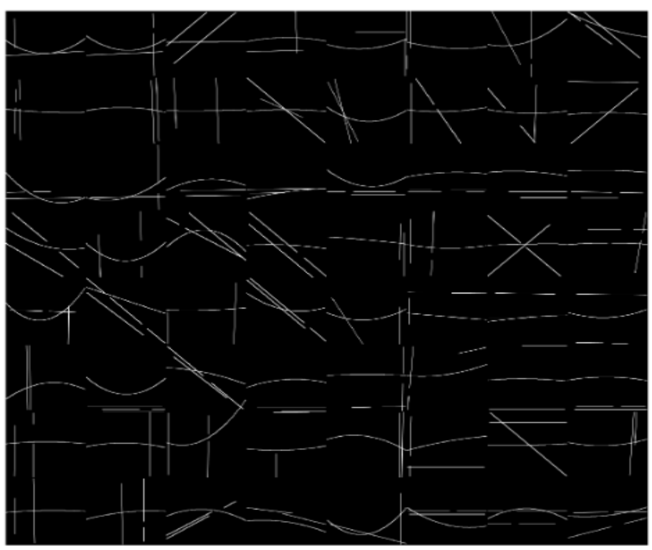

Fig.12. Hough Transform on field.jpg (lines and curves)

DoG Filter –

DoG Filter (Difference of Gaussians) is a filter used in image processing for detecting edges and feature enhancement. The difference between two blurry versions of the same image obtained by applying two different standard deviations of gaussian filter is used to create the DoG filter. So, it works especially well for identifying the areas of an image with sudden changes in intensity. It is observed that such differences in the intensity of an image are found near the edges or borders. In this way, DoG is expected to achieve the required output. So, first the Gaussian blurring is applied to the image with two different values of standard deviation and then the output images are subtracted pixel-wise. This gives us the DoG image and the resulted image has high frequency components. These values are used to identify the edges and borders in the image.

The code begins with reading the input image and the groundtruth image. Firstly, the input image is converted into grayscale image. Later, we tried to calculate the Difference of Gaussians by creating two gaussian filters with different standard deviation values. This DoG filter is then applied to the gray image using convolution and the result is displayed. Additionally, we perform binarization on the DoG filtered image using a particular threshold value. Then alpha bending technique is used inorder to place the binarized image on top of the original gray image. The result is shown in the below picture.

Fig.13. DoG filtered image of L88b.jpg

Fig.14. Binary image of L88b.jpg

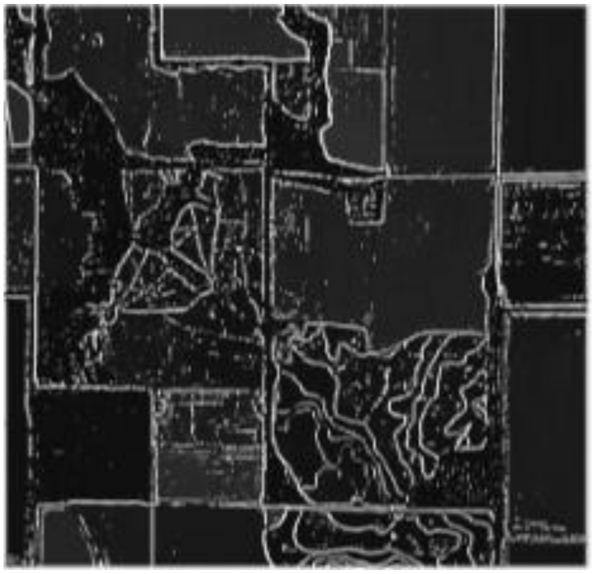

Fig.15. Overlay on Grayscale image for L88b.jpg

3. **Evaluation:** Evaluate the models' performance using different evaluation metrics. The accuracy and precision of the methods is calculated. The result depends on various factors such as image normalization, equalization, noise reduction, and suitable parameters etc. The accuracy of different methods when different parameters are given will also be obtained. The results and findings are discussed in detail in the later part of this document.

QUANTITAVE EVALUATION

Evaluation is one of the most important parts of developing a model. In this project, evaluation of the models' performance is done using evaluation metrics like precision, recall, F1-score, and Intersection over Union (IOU) on the validation dataset, shows how good the developed model is. By evaluating, we can also configure how a model can be improved. So, we have performed some of the evaluation metrics such as accuracy, precision, and SSMI scores for all the models and methods that we used. The values obtained from each method are clearly listed in tabular forms.

The accuracy is calculated using the findIOUAccuracy function in the project implementation. So, this function uses the ground truth image and the predicted image to find the accuracy. For every evaluation metric we try and compare the predicted output by a certain model used in the implementation with the ground truth image that is manually annotated.

NOTE: This accuracy score is calculated by flattening the arrays and comparing the values in the arrays of both images. The accuracies may not be accurate, and this is not the standard method for calculating the accuracies. SSIM is one of the standard methods for finding accuracy. This calculates the black areas as well since it flattens the array not only the predicted lines but also the empty area (black area) is also calculated.

SSIM - The Structural "SIMilarity" (SSIM) index is a method for measuring the similarity between two images. The SSIM index can be viewed as a quality measure of one of the images being compared, provided the other image is regarded as of perfect quality.

Canny Edge Detection

The table below shows the evaluation methodology. It contains the accuracy and precision of Canny Edge detector obtained for each image is listed below.

Name	Size	Type	Filter	Accuracy*	Precision	SSIM	Case
field	2048 x 2048	JPG	Canny Edge Detection	69.33	5.33	47.90	Fail
L88a	2048 x 2048	JPG	Canny Edge Detection	66.97	2.80	33.37	Fail
L88b	2048 x 2048	JPG	Canny Edge Detection	67.20	3.60	35.45	Fail
L96a	2048 x 2048	JPG	Canny Edge Detection	66.07	2.41	39.72	Fail
L96b	2048 x 2048	JPG	Canny Edge Detection	68.25	5.03	42.99	Fail
L97a	2048 x 2048	JPG	Canny Edge Detection	68.63	3.83	46.39	Fail
L97b	2048 x 2048	JPG	Canny Edge Detection	64.40	2.56	24.68	Fail
W107a	2048 x 2048	JPG	Canny Edge Detection	68.16	2.47	38.12	Fail
W107b	2048 x 2048	JPG	Canny Edge Detection	46.89	3.73	36.77	Fail
Average				65.77	3.42	38.38	Fail

Table 3. Accuracy and Precision Table for Canny Edge Detection

Sobel Edge Detection

The table below shows the evaluation methodology. It contains the accuracy and precision of Sobel Edge detector obtained for each image is listed below.

Name	Size	Type	Filter	Accuracy*	Precision	SSIM	Case
field	2048 x 2048	JPG	Sobel Edge Detection	64.63	14.37	64.87	Fail
L88a	2048 x 2048	JPG	Sobel Edge Detection	67.06	8.65	64.61	Fail
L88b	2048 x 2048	JPG	Sobel Edge Detection	75.39	11.25	62.52	Fail
L96a	2048 x 2048	JPG	Sobel Edge Detection	72.85	3.80	51.27	Fail
L96b	2048 x 2048	JPG	Sobel Edge Detection	75.44	15.76	77.19	
L97a	2048 x 2048	JPG	Sobel Edge Detection	75.14	8.56	67.96	Fail

L97b	2048 x 2048	JPG	Sobel Edge Detection	73.76	6.42	48.66	Fail
W107a	2048 x 2048	JPG	Sobel Edge Detection	60.22	10.46	69.98	Fail
W107b	2048 x 2048	JPG	Sobel Edge Detection	44.73	11.84	65.67	Fail
Average				67.69	11.26	63.64	Fail

Table 4. Accuracy and Precision Table for Sobel Edge Detection

Harris Corner Detection

The table below shows the evaluation methodology. It contains the accuracy and precision of Harris Corner detector obtained for each image listed below.

Name	**Size**	**Type**	**Filter**	**Accuracy***	**Precision**	**SSIM**	**Case**
field	2048 x 2048	JPG	Harris Corner Detection	66.127	10.44	59.4	Fail
L88a	2048 x 2048	JPG	Harris Corner Detection	65.69	5.94	45.95	Fail
L88b	2048 x 2048	JPG	Harris Corner Detection	66.64	12.49	50.22	Fail
L96a	2048 x 2048	JPG	Harris Corner Detection	65.16	2.89	41.07	Fail
L96b	2048 x 2048	JPG	Harris Corner Detection	65.68	11.74	60.94	Fail
L97a	2048 x 2048	JPG	Harris Corner Detection	66.39	7.55	57.64	Fail
L97b	2048 x 2048	JPG	Harris Corner Detection	63.61	3.51	32.93	Fail
W107a	2048 x 2048	JPG	Harris Corner Detection	68.19	6.25	66.57	Fail
W107b	2048 x 2048	JPG	Harris Corner Detection	66.47	12.74	58.77	Fail
Average				66.65	8.17	52.61	Fail

Table 5. Accuracy and Precision Table for Harris Corner Detection

DoG filter

The table below shows the evaluation methodology. It contains the accuracy and precision of DoG filter obtained for each image listed below.

Name	**Size**	**Type**	**Filter**	**Accuracy***	**Precision**	**SSIM**	**Case**
field	2048 x 2048	JPG	DoG	81.82	14.21	76.8	Pass
L88a	2048 x 2048	JPG	DoG	81.21	8.87	69.10	Pass
L88b	2048 x 2048	JPG	DoG	79.16	11.32	64.82	Fail

L96a	2048 x 2048	JPG	DoG	81.40	9.70	71.84	Pass
L96b	2048 x 2048	JPG	DoG	81.59	15.64	77.94	Pass
L97a	2048 x 2048	JPG	DoG	80.61	9.83	72.50	Pass
L97b	2048 x 2048	JPG	DoG	79.47	8.05	61.54	Fail
W107a	2048 x 2048	JPG	DoG	84.13	9.41	82.61	Pass
W107b	2048 x 2048	JPG	DoG	80.27	14.72	70.471	Pass
Average				81.07	11.31	71.95	Pass

Table 6. Accuracy and Precision Table for DoG filter

Hough Transform (Straight lines only)

The table below shows the evaluation methodology. It contains the accuracy and precision of Hough Transform obtained for each image listed below.

Name	**Size**	**Type**	**Filter**	**Accuracy***	**Precision**	**SSIM**
field	2048 x 2048	JPG	Hough Transform	86.44	12.94	80.54
L88a	2048 x 2048	JPG	Hough Transform	86.55	5.23	77.97
L88b	2048 x 2048	JPG	Hough Transform	86.36	6.69	76.58
L96a	2048 x 2048	JPG	Hough Transform	86.46	6.93	80.19
L96b	2048 x 2048	JPG	Hough Transform	86.04	11.93	85.47
L97a	2048 x 2048	JPG	Hough Transform	86.71	6.78	85.96
L97b	2048 x 2048	JPG	Hough Transform	86.32	5.47	73.09
W107a	2048 x 2048	JPG	Hough Transform	87.11	6.39	86.96
W107b	2048 x 2048	JPG	Hough Transform	86.39	9.41	83.95
Average				85.98	7.97	83.52

Table 7. Accuracy and Precision Table for Hough Transform (lines)

Hough Transform (curves)

The table below shows the evaluation methodology. It contains the accuracy and precision of Hough Transform detecting both lines and curves obtained for each image listed below.

Name	Size	Type	Filter	SSIM	Precision	Accuracy*
field	2048 x 2048	JPG	Hough Transform	80.85	9.19	85.62
L88a	2048 x 2048	JPG	Hough Transform	77.97	4.13	85.74
L88b	2048 x 2048	JPG	Hough Transform	76.58	5.33	85.50
L96a	2048 x 2048	JPG	Hough Transform	80.19	5.48	85.65
L96b	2048 x 2048	JPG	Hough Transform	85.47	8.9	85.26
L97a	2048 x 2048	JPG	Hough Transform	85.96	5.19	85.88
L97b	2048 x 2048	JPG	Hough Transform	73.09	4.40	85.53
W107a	2048 x 2048	JPG	Hough Transform	86.96	4.05	86.32
W107b	2048 x 2048	JPG	Hough Transform	83.95	6.74	85.57
Average				81.19	4.80	85.67

Table 7. Accuracy and Precision Table for Hough Transform (curves)

OUTCOME

In this part of the report, we attempted to display all possible output images obtained for all the methods that we implemented in the project.

Canny Edge Detector

The detection of field and road boundaries in the given dataset obtained after applying the canny edge detection are given below:

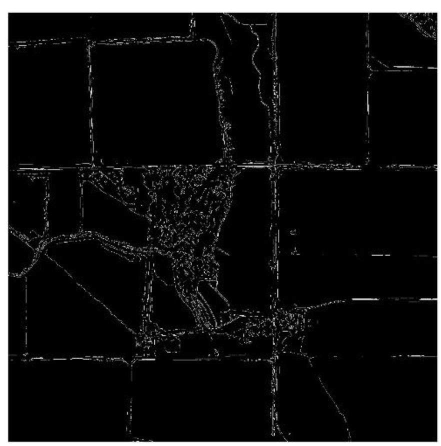

Fig 16. Canny on Field.jpg

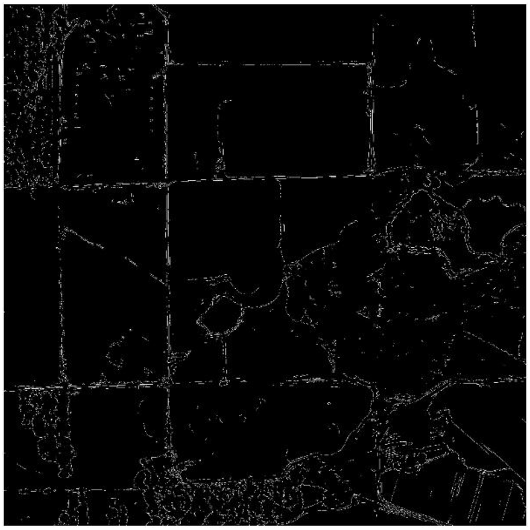

Fig 17. Canny on L88a.jpg

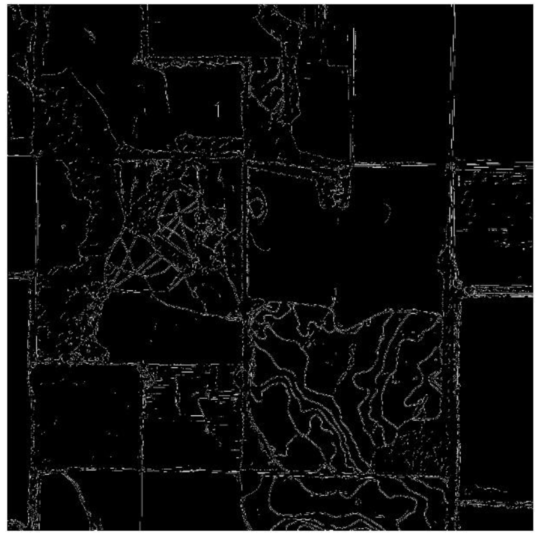

Fig 18. Canny on L88b.jpg

Fig 19. Canny on L96a.jpg

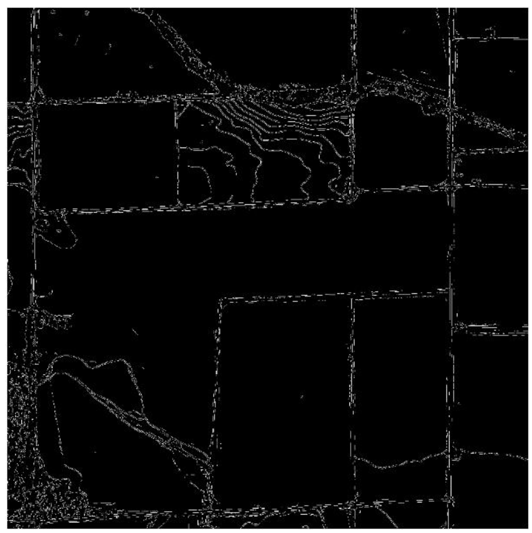

Fig 20. Canny on L96b.jpg

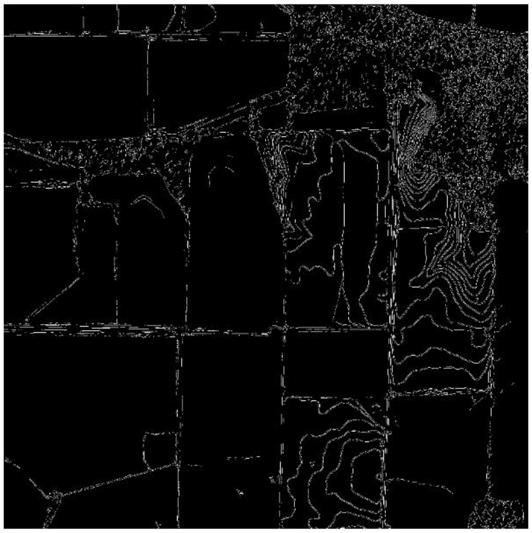
Fig 21. Canny on L97a.jpg

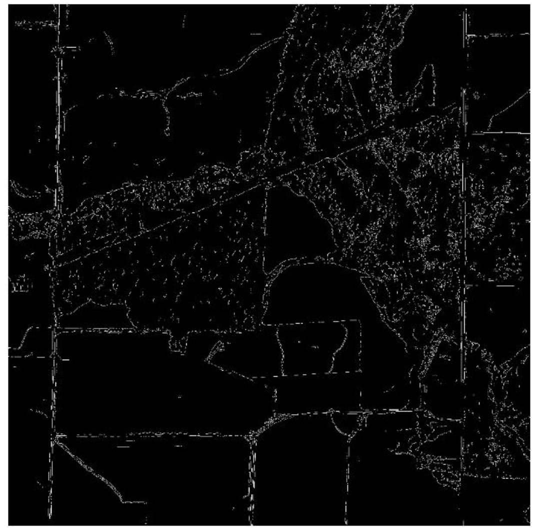
Fig 22. Canny on L97b.jpg

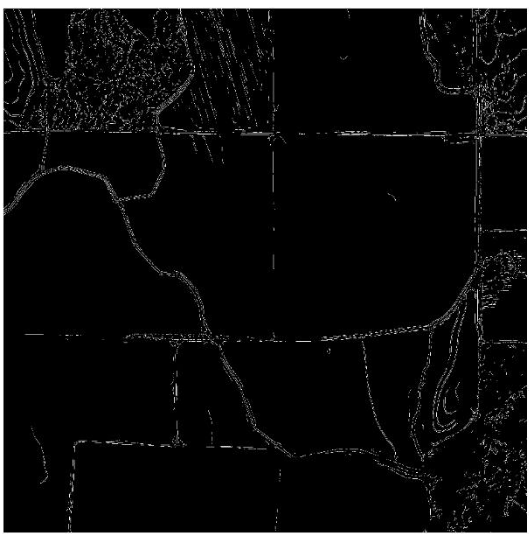
Fig 23. Canny on W107a.jpg


Fig 24. Canny on W107b.jpg

Sobel Edge Detector

The detection of field and road boundaries in the given dataset obtained after applying the Sobel edge detection are given below:

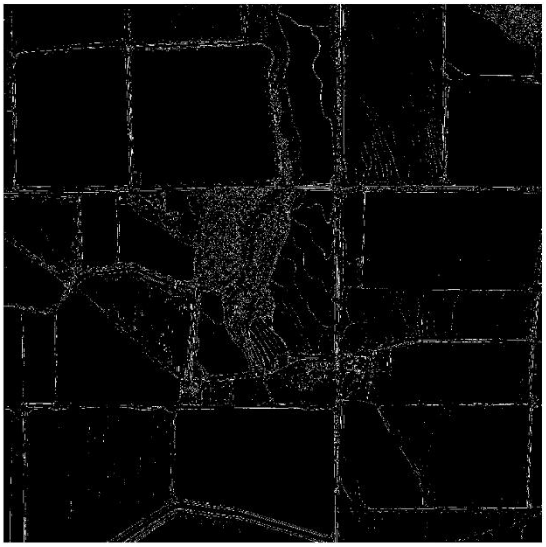

Fig 25. Sobel on Field.jpg

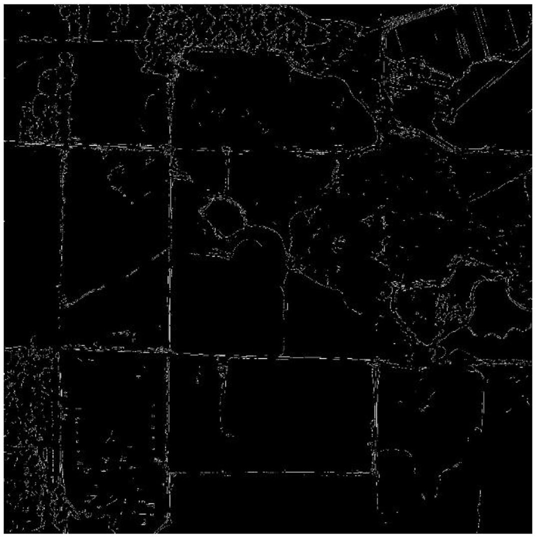

Fig 26. Sobel on L88a.jpg

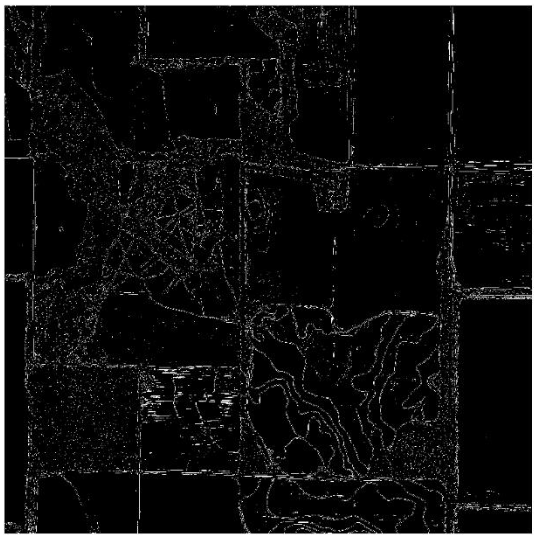

Fig 27. Sobel on L88b.jpg


Fig 28. Sobel on L96a.jpg

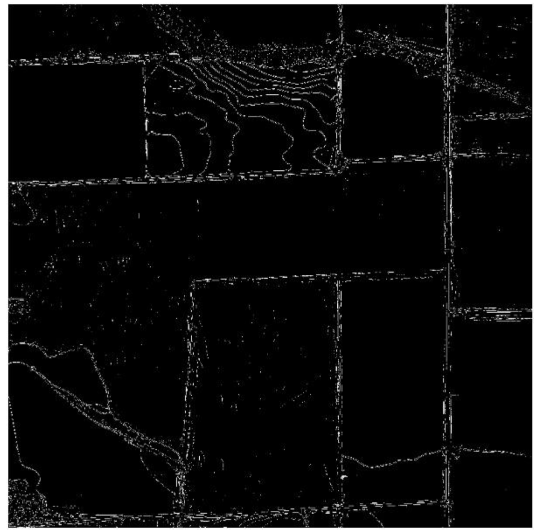
Fig 29. Sobel on L96b.jpg

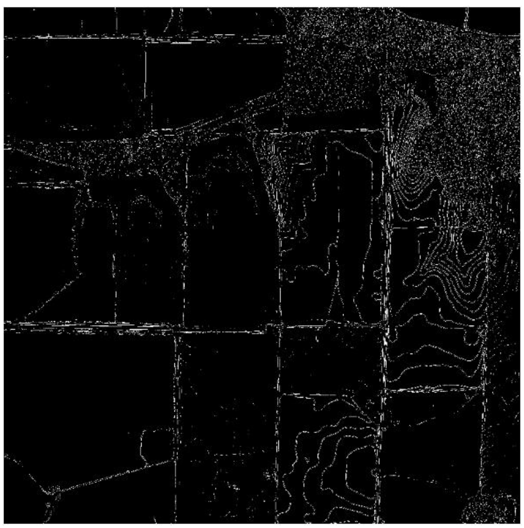
Fig 30. Sobel on L97a.jpg

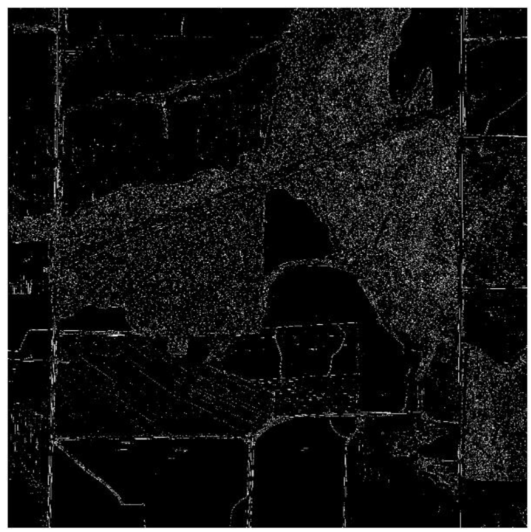
Fig 31. Sobel on L97b.jpg

Fig 32. Sobel on W107a.jpg

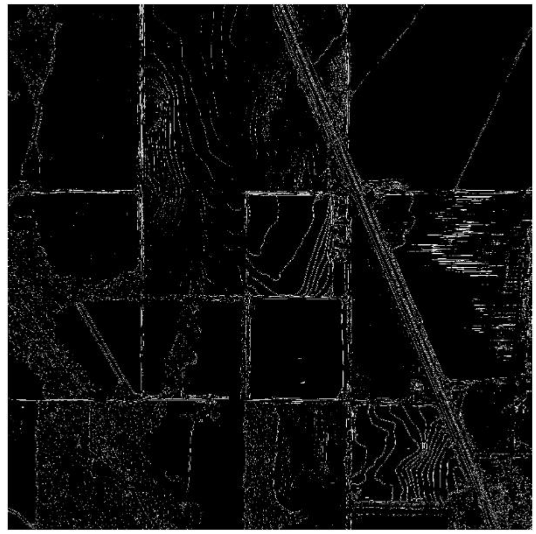

Fig 33. Sobel on W107b.jpg

Harris Corner Detector

The detection of field and road boundaries in the given dataset obtained after applying the Harris Corner edge detection are given below:

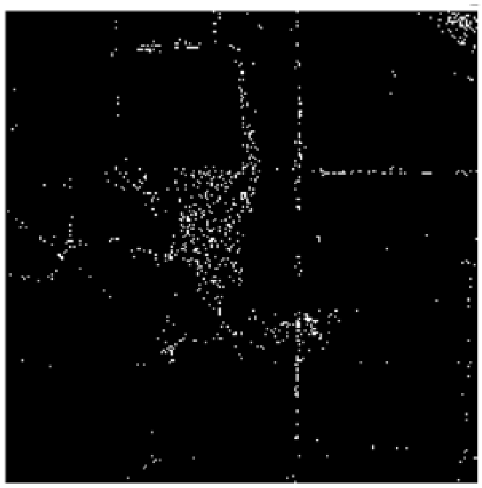

Fig 34. Harris Corner on Field.jpg

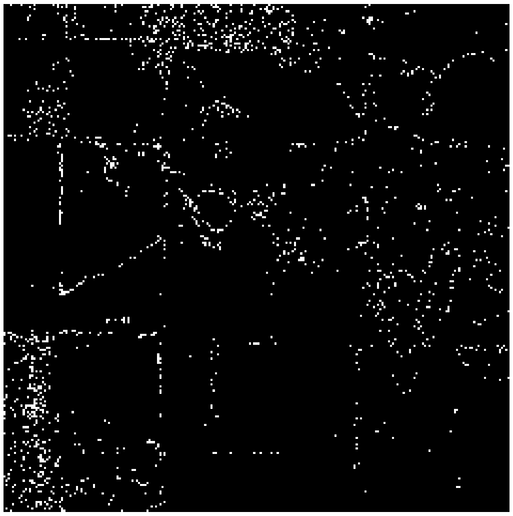

Fig 35. Harris Corner on L88a.jpg

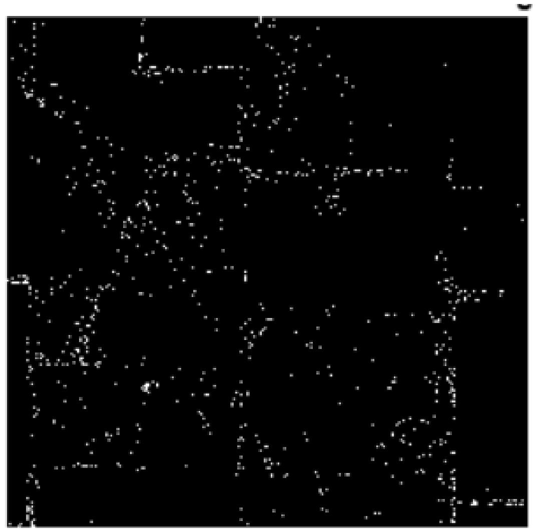

Fig 36. Harris Corner on L88b.jpg

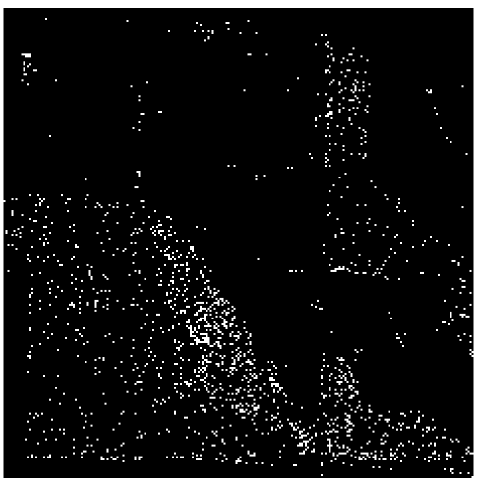

Fig 37. Harris Corner on L96a.jpg

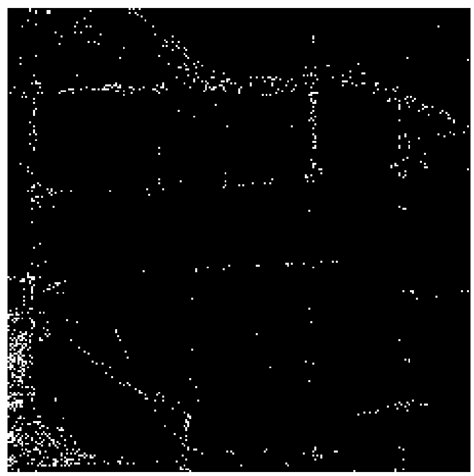

Fig 38. Harris Corner on L96b.jpg

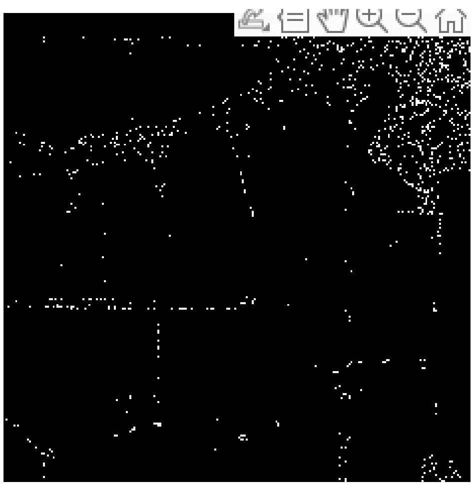

Fig 39. Harris Corner on L97a.jpg

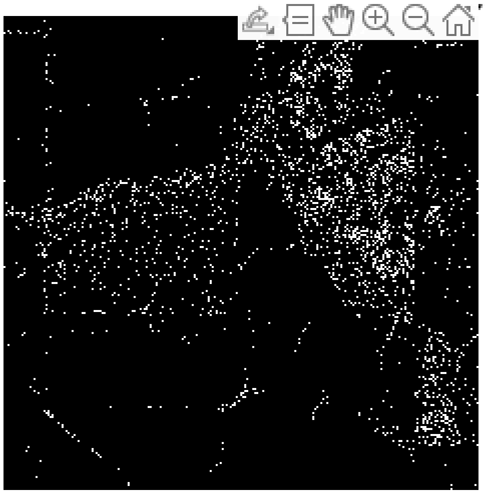

Fig 40. Harris Corner on L97b.jpg

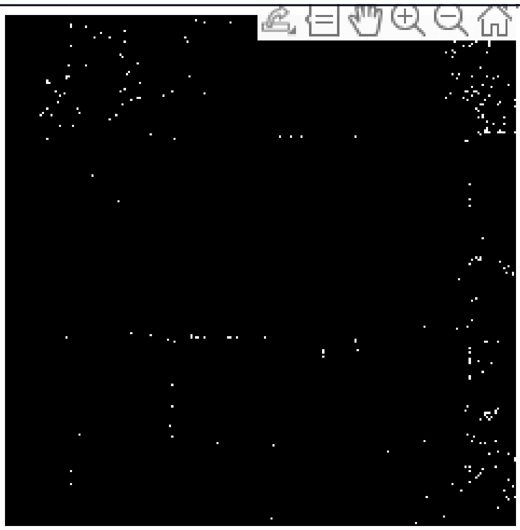

Fig 41. Harris Corner on W107a.jpg

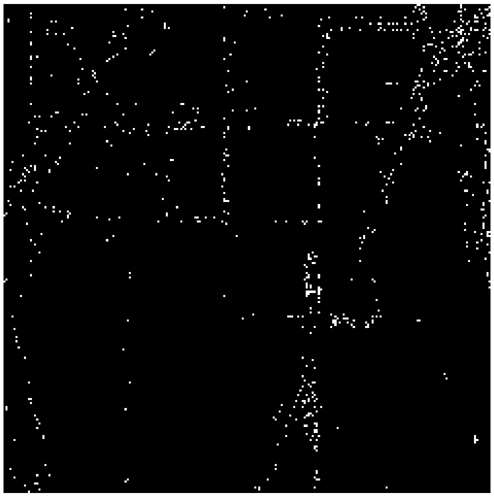

Fig 42. Harris Corner on W107b.jpg

DoG filter

The detection of field and road boundaries in the given dataset obtained after applying the DoG filter are given below:

Fig 43. DoG on Field.jpg

Fig 44. DoG on L88a.jpg

Fig 45. DoG on L88b.jpg


Fig 46. DoG on L96a.jpg


Fig 47. DoG on L96b.jpg

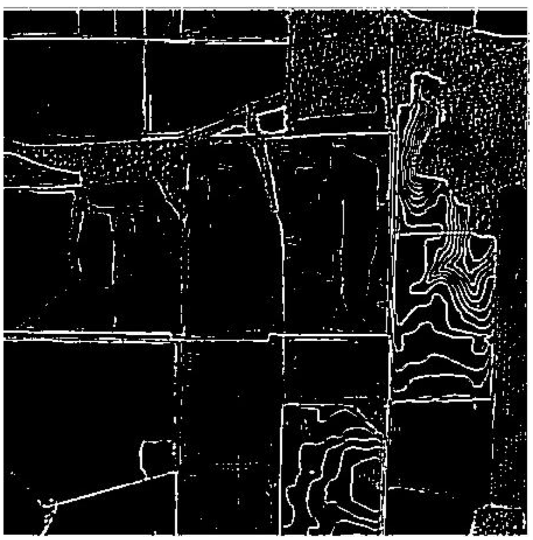
Fig 48. DoG on L97a.jpg


Fig 49. DoG on L97b.jpg

Fig 50. DoG on W107a.jpg

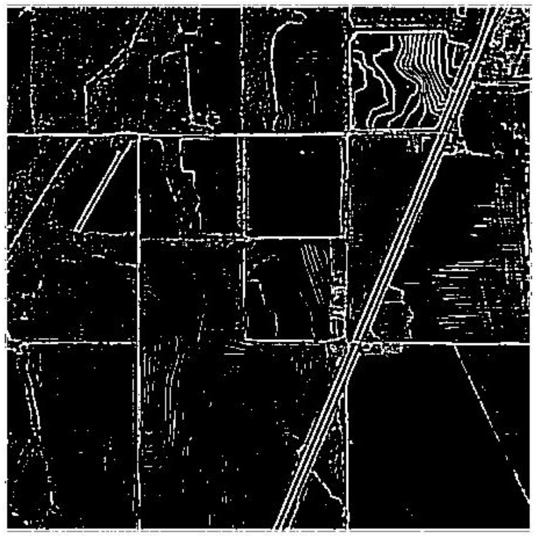

Fig 51. DoG on W107b.jpg

Hough Transform with Curves.

The detection of field and road boundaries in the given dataset obtained after applying the Hough Transform are given below:

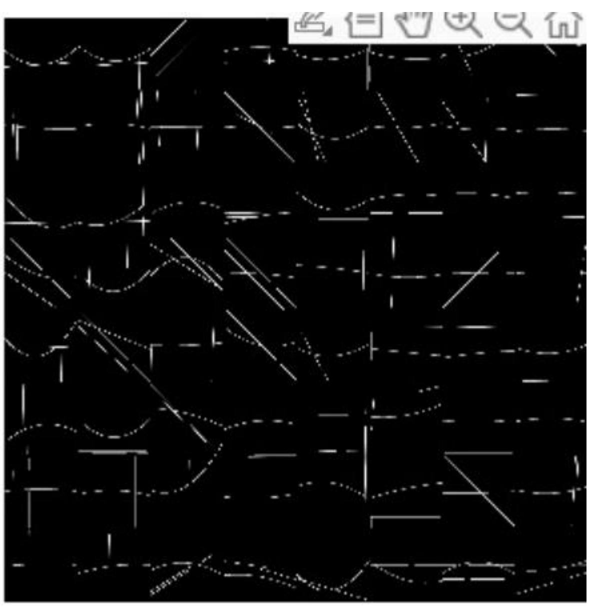

Fig 52. Hough Transform on Field.jpg

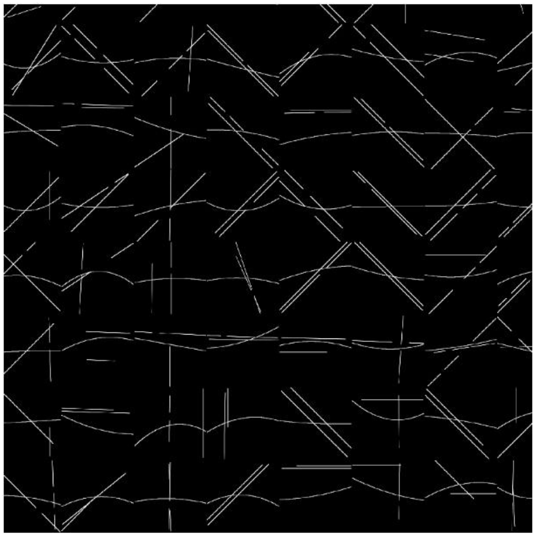

Fig 53. Hough Transform on L88a.jpg

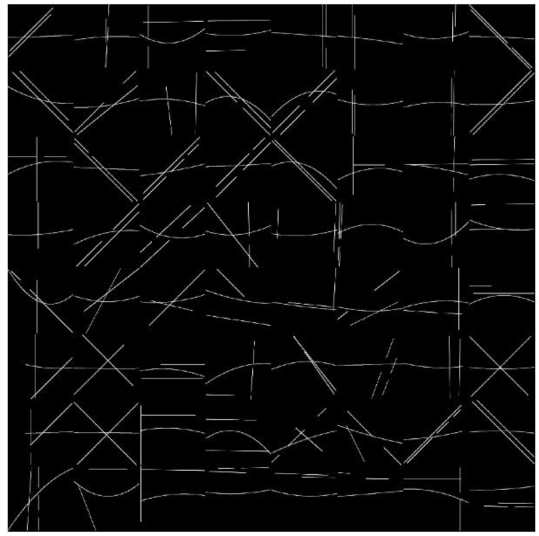

Fig 54. Hough Transform on L88b.jpg

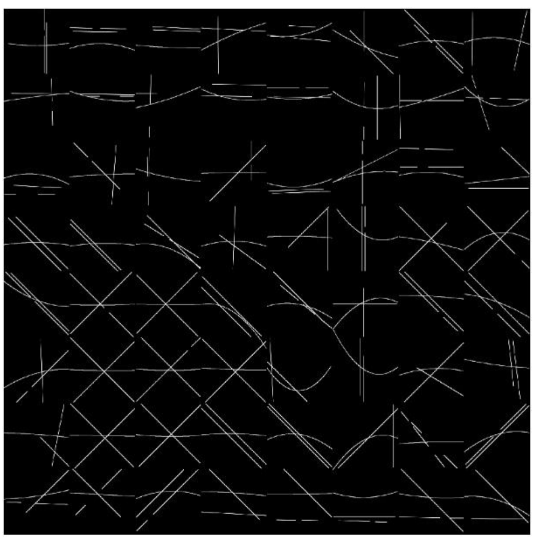

Fig 55. Hough Transform on L96a.jpg

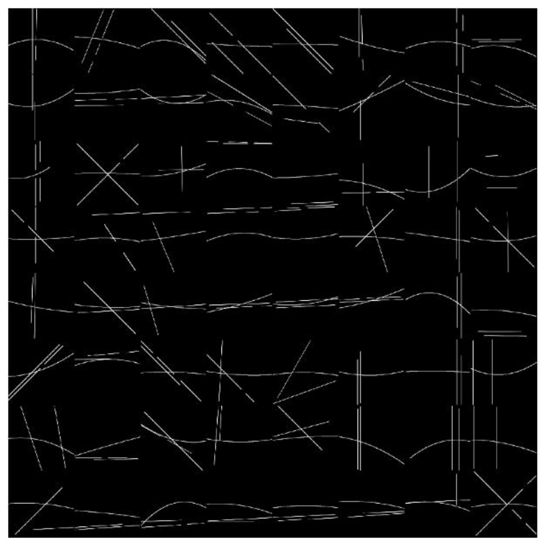

Fig 56. Hough Transform on L96b.jpg

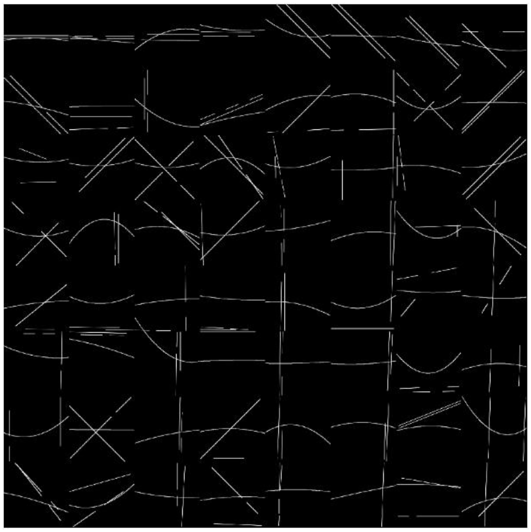
Fig 57. Hough Transform on L97a.jpg

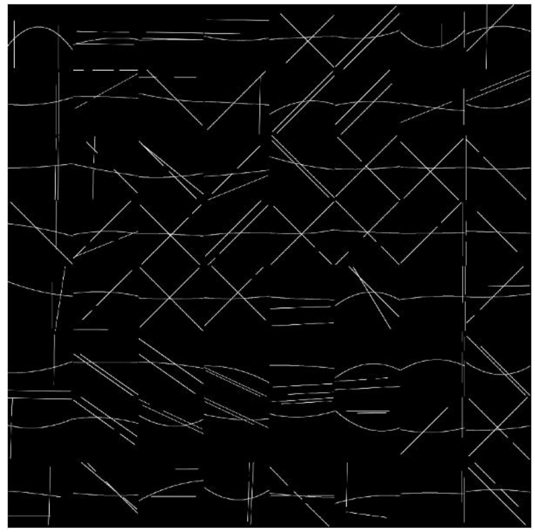
Fig 58. Hough Transform on L97b.jpg

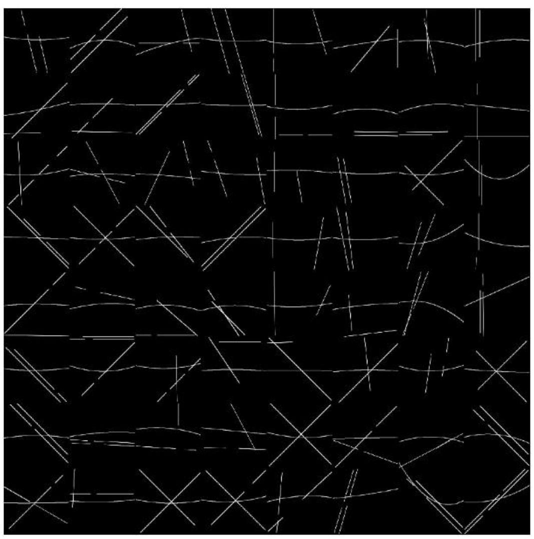
Fig 59. Hough Transform on W107a.jpg

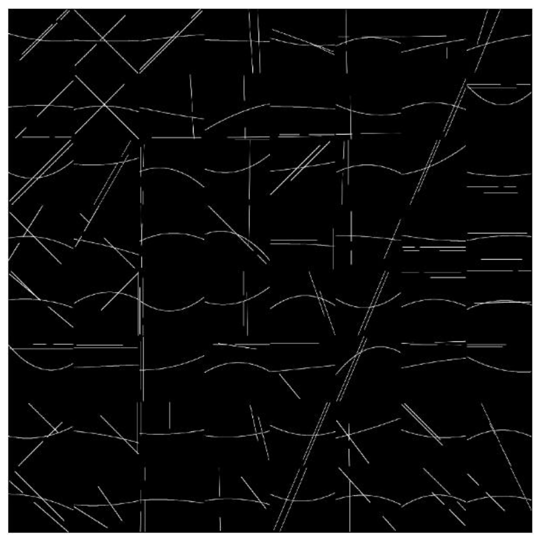
Fig 60. Hough Transform on W107b.jpg

Ground Truth Images

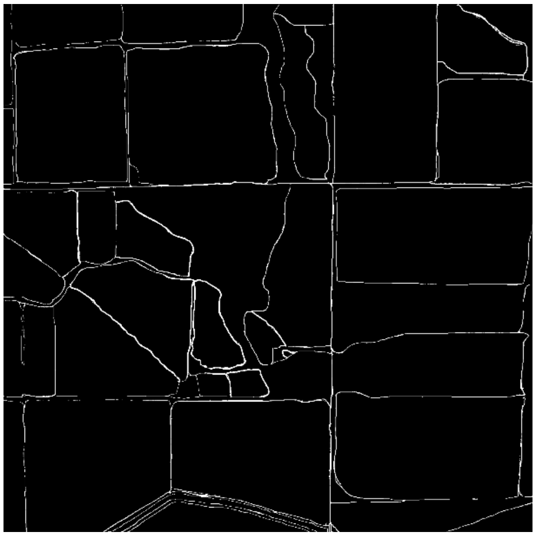

Fig 61. Ground Truth of Field.jpg

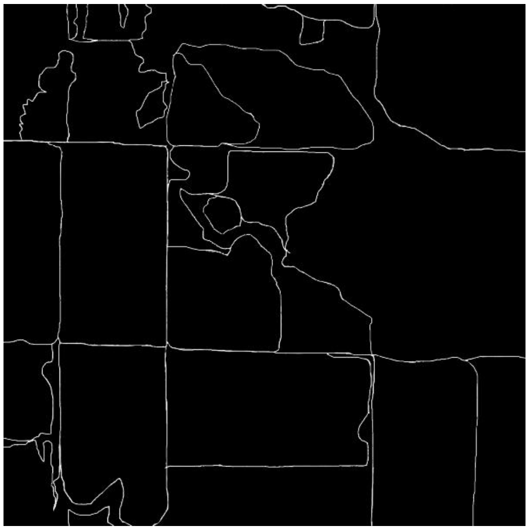

Fig 62. Ground Truth of L88a.jpg

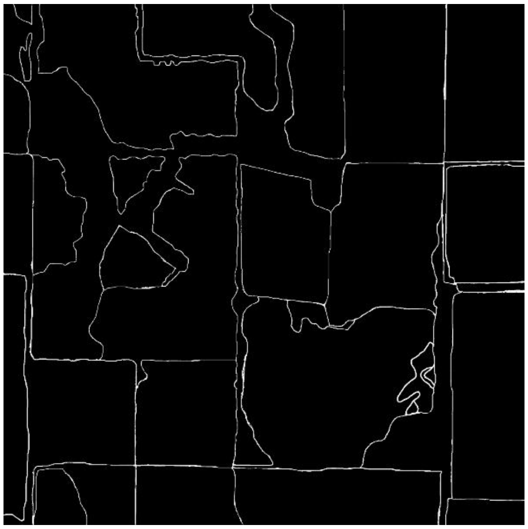

Fig 63. Ground Truth of L88b.jpg


Fig 64. Ground Truth of L96a.jpg

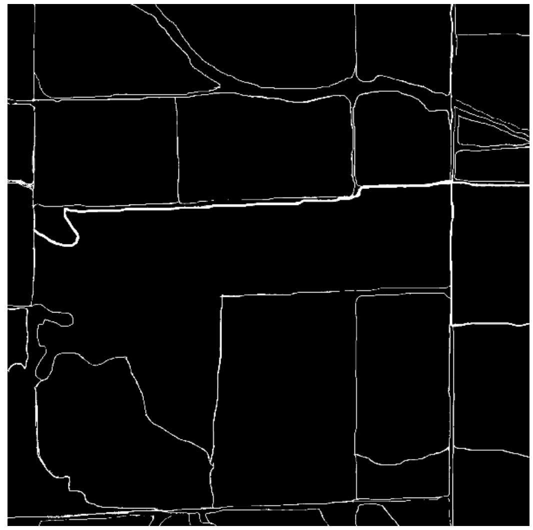
Fig 65. Ground Truth of L96b.jpg


Fig 66. Ground Truth of L97a.jpg

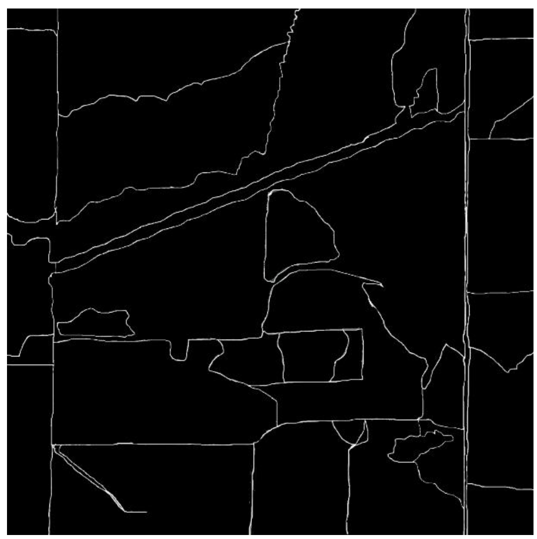
Fig 67. Ground Truth of L97b.jpg

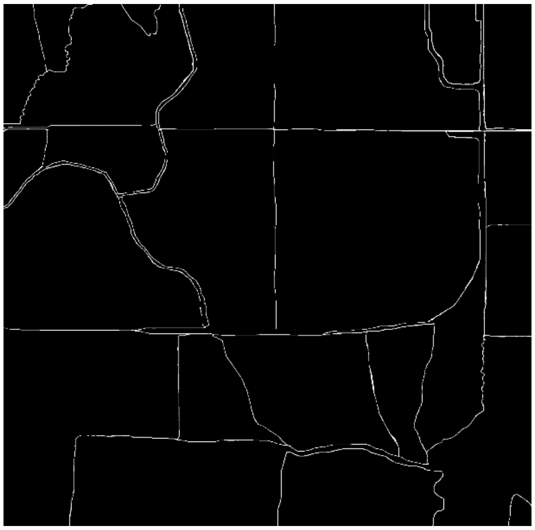

Fig 68. Ground Truth of W107a.jpg

Fig 69. Ground Truth of W107b.jpg

RESULT

From all the methods like Canny, Sobel with many variations in thresholds, Harris Corner, Hough Transform, Hough Transform with splitting, DOG Filter that we have developed and tested. From the results we can see that DoG is more consistent and produces better results when compared to all the other models.

Consistency can be seen by comparing the averages and the individual values generated. The difference from means is very low when compared to all the other methods.

Method Applied	**SSIM**	**Accuracy***	**Precision**
Sobel	63.64	67.69	11.26
Canny	38.38	65.77	3.42
Harris Corner Detection	52.61	66.65	8.17
Hough Straight Lines only	83.52	85.98	7.97
Hough with curves	81.19	85.67	4.80
Dog Filter	71.95	81.07	11.31

Table 8. Results Table (Averages Only)

Models like Canny and Sobel are Fail cases where the detection is not up to the mark as we have expected. The detection is not consistent.

DISCUSSION

We carefully annotated ground truth images and used them to thoroughly evaluate the accuracy of our image processing techniques at each step. A benchmark average accuracy of 68% was set to judge success(Considering SSIM scores only). Capturing sufficient detail in roads and field boundaries was another criterion. The Structural Similarity Index (SSIM) allowed quantitative comparisons to ground truth.

Initially, distinguishing between forests colors and grounds posed challenges for conventional filters and edge detection. We moved to using nonstandard operations instead.

Given closing deadlines, we applied Gaussian filtering using the fspecial function. We also newly introduced Difference of Gaussians (DoG) filtering after reviewing relevant documentation.

Revisiting DoG filtering yielded better boundary detection. Adjusting parameters by testing on images further improved result.

Converting the resultant images to binary validated the anticipated output. Consistent outputs across diverse image sets highlighted the reliability of the DoG filtering approach.

We have considered DoG filter as our main because of its consistency in detecting curves. We can see that the SSIM scores of Hough straight lines and curves are higher but the reason we donot consider is due to the reason for detection of curved roads and boundaris.

CONCLUSION

By using Gaussian and difference of Gaussian and converting the result image to binary we have brought our image close to ground truth and also if there was still any time we would like to perform some other operations like using Harris corner detection with Hough Transform splitting Technique on the image.

EXPECTED OUTCOME

List of activities	Expected Outcome
Project Kickoff and planning	Begin the project
Data acquisition and initial preprocessing	Dataset gathering Dataset preprocessing by applying filters to remove noise, Histogram Stretching, Equalization, Median Filtering etc.
Begin model selection and Ground Truth setup	Performing manual annotations to create ground truth for the dataset. Models' selection such as Sobel edge detectors, Canny edge detectors etc.
Continue model selection and setup	Ground Truth images. Working models of edge detectors to achieve project goal. Application of models on the ground truth images.
Try different hyperparameters to improve performance.	By applying different thresholds for different methods to improve models' performances.
Complete the model.	Final models should be able to detect the lines along with points detection. Hough Transform.
Evaluate the model	Models' accuracy and precision calculation. Compare the models.
Report	Problem Statement, Methods Description, Evaluation, Results, Models' comparison, Future Scope.

CONTRIBUTION to the WORK

List of activities	Timeline	Team Member
Project Kickoff and planning		Worked together.

Data acquisition and initial preprocessing.		Harsha, Sasidhar
Begin model selection and setup.		Harsha, Preethi
Continue model selection and setup.		Harsha, Preethi
Try different hyperparameters to improve performance.	Oct 1 – Oct 15	Worked together.
Complete the model.	Oct 16 – Oct 30	Worked together.
Evaluate the model.	Nov 1 – Nov 15	Worked together.
Final Implementation and Report	Nov 15 – Dec 2	Worked together.
Final Presentation PPT	Nov 15 – Dec 4	Worked together.

Table.4. Project Timeline

REFERENCES:

[1] Apeer.com https://www.apeer.com/app/dashboard.

[2] https://stackoverflow.com/questions/39123267/how-to-detect-smooth-curves-in-matlab

[3] Matlab Documentation