Road and Field Boundary Detection in Satellite Imagery

Group 15

Harshavardhan Reddy Mallannagari, Preethi Vahitha Sankineni, Sasidhar Yalamanchili

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.

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 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.

Fig.1. field.jpg

Method

For the given project to be successful we need to perform various image processing methods, edge detection techniques and accuracy evaluations. A detailed explanation on the same is discussed here. They are mentioned below:

1. **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.

2. **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].

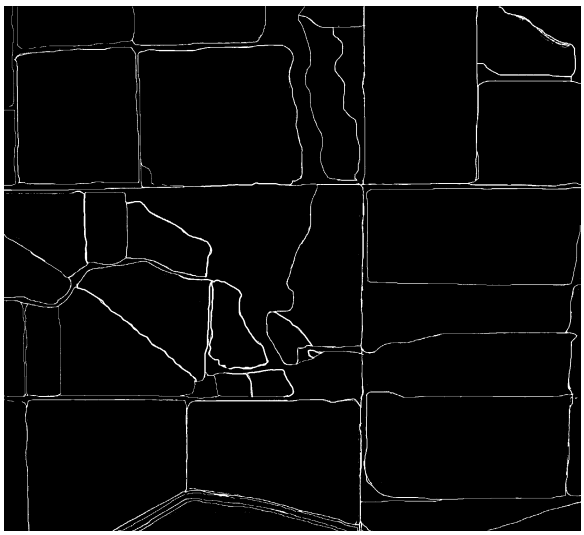

Fig.2. Ground Truth of field.jpg

3. **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. 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.4. Binary image of field.jpg

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 have 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.

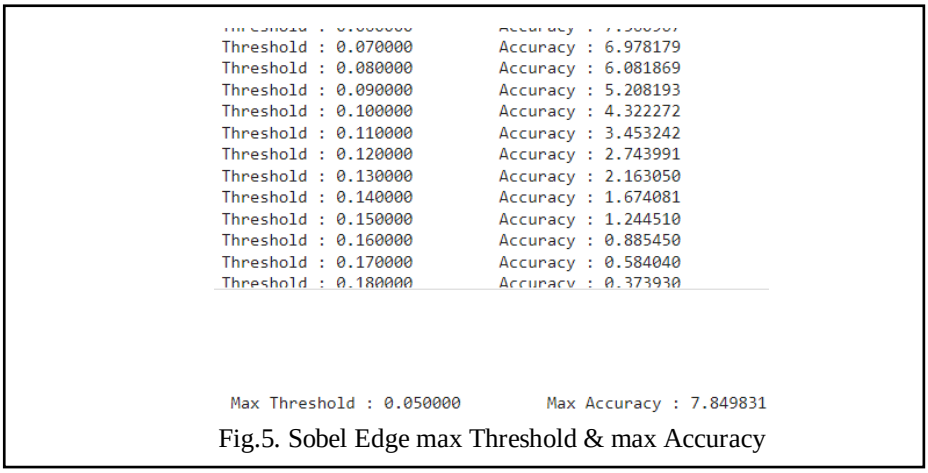

Fig.5. Sobel Edge max Threshold & max Accuracy

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**	**Accuracy**
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:

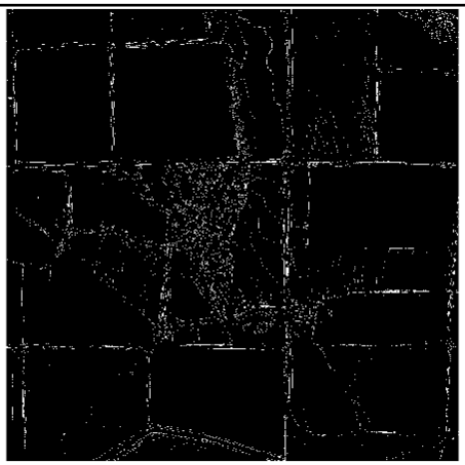
Fig.6. Sobel Edge Detector for field.jpg

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 filter is applied to the image. The accuracy of Canny Edge Detection 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.

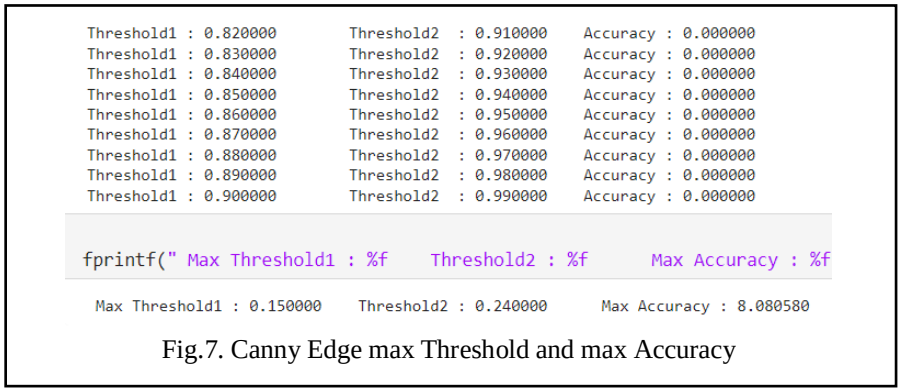

Fig.7. Canny Edge max Threshold and max Accuracy

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**	**Accuracy**
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:

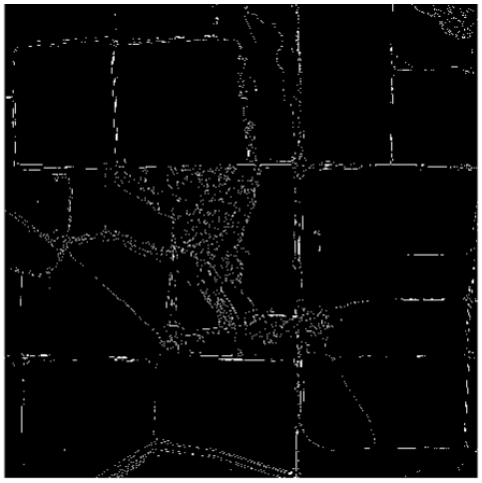

Fig.8 Canny Edge Detector for field.jpg

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.

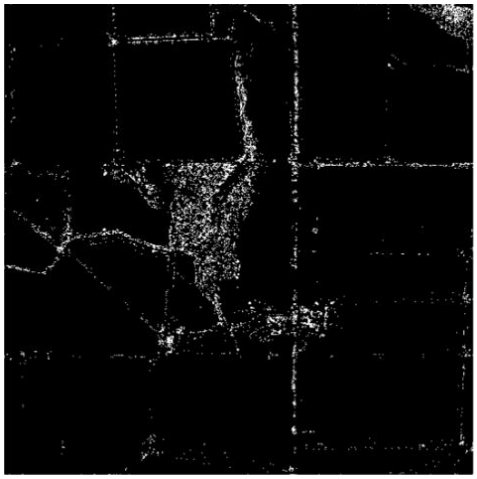

Fig.9 Harris Corner Detection on field.jpg

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.

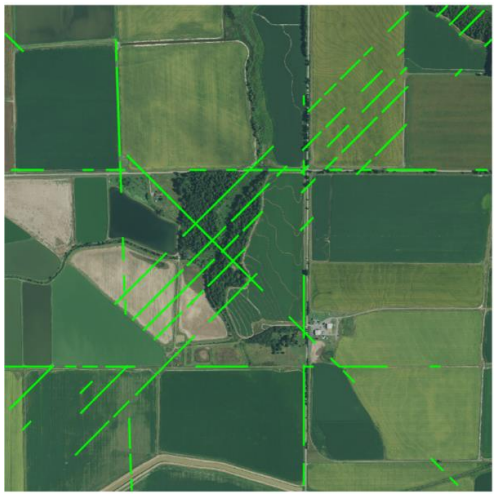

Fig.10. Virtual Lines in Hough Transform on field.jpg

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 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. We performed Hough transform on the L88a.jpg image and the result is shown below.

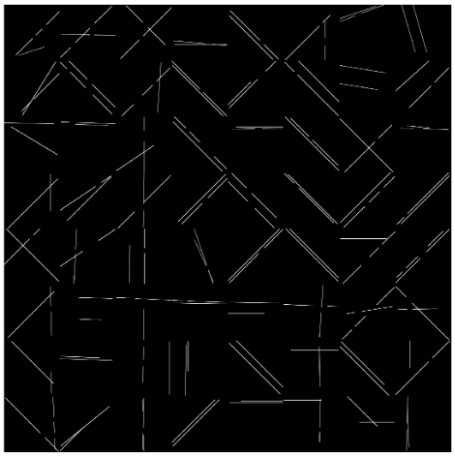

Fig.11. Hough Transform on L88a.jpg (lines)

While performing Hough Transform for the second time, 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.

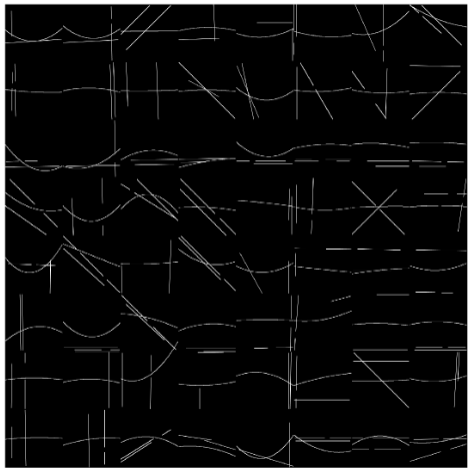

Fig.12. Hough Transform on field.jpg (lines and curves)

DoG Filter –

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.

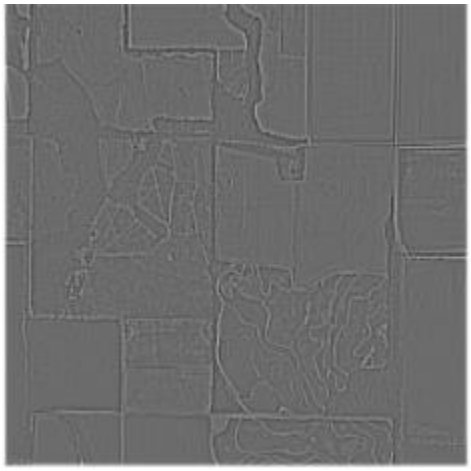

Fig.13. DoG filtered image of L88b.jpg

Fig.14. Binary image of L88b.jpg

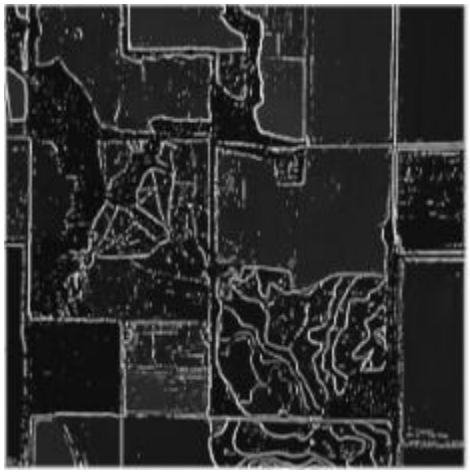

Fig.15. Overlay on Grayscale image for L88b.jpg

4. **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.

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.

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

Name	Size	Type	Filter	Accuracy	Precision
field	2048 x 2048	JPG	DoG, Hough Transform	76.8, 80.54	Pending
L88a	2048 x 2048	JPG	DoG, Hough Transform	69.10, 77.97	Pending
L88b	2048 x 2048	JPG	DoG, Hough Transform	64.82, 76.58	Pending
L96a	2048 x 2048	JPG	DoG, Hough Transform	71.84, 80.19	Pending
L96b	2048 x 2048	JPG	DoG, Hough Transform	77.94, 85.47	Pending
L97a	2048 x 2048	JPG	DoG, Hough Transform	72.50, 85.96	Pending
L97b	2048 x 2048	JPG	DoG, Hough Transform	61.54, 73.09	Pending
W107a	2048 x 2048	JPG	DoG, Hough Transform	82.61, 86.96	Pending
W107b	2048 x 2048	JPG	DoG, Hough Transform	70.471, 83.95	Pending

Table.3. Evaluation Metrics

Current Work Contribution

List of activities	Timeline	Team Member
Project Kickoff and planning	Sep 15 – Sep 30	Worked together.
Data acquisition and initial preprocessing.		Harsha, Sasidhar
Begin model selection and setup.		Harsha, Preethi
Continue model selection and setup.	Oct 1 – Oct 15	Harsha, Preethi
Try different hyperparameters to improve performance.		Worked together.
Complete the model.	Oct 16 – Oct 30	Worked together.
Evaluate the model.	Nov 1 – Nov 15	Worked together.

Table.4. Project Timeline

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.

Table.5. Expected Project Outcome

TimeLine for Future Work

Methods	Timeline	Team Members
Implementing Harris & Hough	Nov 16 – Nov 30	Working together
Gabor Filter (if time permits)	Nov 16 – Nov 30	Working together
Hough + DoG	Nov 16 – Nov 30	Working together
Full Report	Nov 16 – Nov 30	Working together
Overall Presentation	Nov 16 – Nov 30	Working together

Table.6. Future Work

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