Road Detection using Satilite Images

Group 13

Shanmukha Datta Kogatni, Prasaad Reddy Nandi, Izzaz Ali

1 Abstract:

This project addresses the segmentation challenge using a dataset of aerial images provided by the USDA, focusing on discerning roads and field boundaries. We apply several segmentation techniques, scrutinizing their performance against a set of curated evaluation metrics. The objective is not merely to delineate crop fields but to segment the images based on line features that define roads and boundaries, despite their inconsistent visual characteristics. Through methodical experimentation with nine distinct images, our research identifies the most effective algorithms for our specific use case.

2 Introduction:

Within the broad field of computer vision, segmentation is a fundamental idea that is essential to distinguishing the target area of interest from its surroundings in an image. In this research, nine satellite photos from the U.S. Department of Agriculture (USDA) are segmented to identify distinguishable elements. Large areas of land are revealed in these high-resolution photos, which depict a range of topographies. We focus on the complex designs of agricultural landscapes, which are distinguished by distinct colour and texture, from the vivid greens of the crops to the muted browns of the fallow area. The fine lines of field partitions and highways, which move across the landscape in both a noticeable and subtle presence, are interspersed throughout these agricultural mosaics. Additionally, the images includes areas with residential buildings, irregular groups of trees, and the placid blue of bodies of water, each with distinct edges and outlines.

3 Problem Statement:

Our project's objective is to precisely divide road networks using satellite photos, which presents a difficult problem because of the uneven topography. The USDA donated the photographs, which show a variety of rural settings with highways blending into natural and farmed regions, sometimes with no obvious boundaries. The objective is to create an algorithm that can identify these roads,

which differ in width, colour, and continuity, against a variety of backgrounds including water features, vegetation, and crop fields. In order to support autonomous car technology and improve navigational assistance, this division is essential.

4 Methodology:

To address the segmentation of roads from the provided satellite images, our methodology integrates a suite of seven sophisticated techniques that leverage the strengths of Sobel, Canny, and Laplacian edge detection algorithms.

Figure 1: Original Image

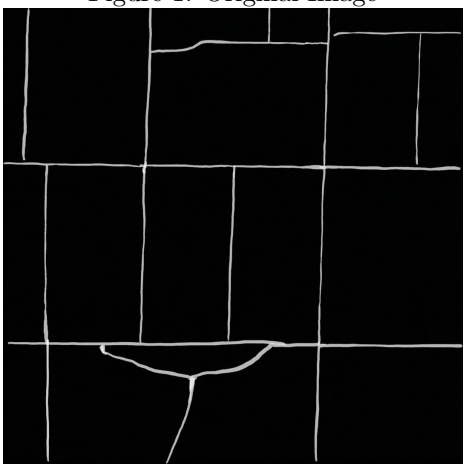

Figure 2: Ground Truth Image

4.1 Sobel Edge Detection:

The Gradient-based Sobel Operator is a fundamental method for identifying road networks in satellite data. By calculating the gradient magnitude at each point in the image, this operator highlights the high spatial frequency regions that usually correlate to edges. Road boundaries are represented comprehensively by applying the Sobel operator independently along the x and y axes. These transitions are indicative of the road boundaries against the comparatively consistent textures of neighbouring fields. After generating an edge-mapped image, possible highways are highlighted and subsequently separated from other linear features by additional thresholding. Even when visibility varies because of shifting agricultural activity or other natural phenomena, this method is especially good at capturing the structural integrity of roads. Our multifaceted analytical approach begins with the creation of a strong initial road segmentation model, made possible by the sensitivity of the Sobel Operator to edge direction.

Since edge detection in the Sobel-edge Detection process does not require colour information, we first prepare the image by converting it to grayscale. The grayscale image is then normalised to improve contrast and make it easier to identify the edges. We use a Contrast Limited Adaptive Histogram Equalisation (CLAHE) strategy to further improve edge visibility and refine image quality. This technique adjusts the image histogram to boost local contrast, which is especially useful in areas with lower contrast or darker areas. Pre-processing is an important step because it lays the groundwork for the Sobel operator to recognise edges with greater accuracy.

The equalised image is then subjected to a Gaussian Blur to reduce noise and prevent erroneous edge detection. After that, edge detection is carried out in both the horizontal and vertical axes using the Sobel operator, producing two distinct matrices of edge data. These edges' magnitudes are calculated to provide an output of one edge, which is then normalised to increase visibility. We dilate the margins to highlight them and make them stand out more for the thresholding phase that follows. We determine the ideal threshold to transform the edge data into a binary image using Otsu's approach. Morphological operations (closing and opening) are used to smooth out the edges and lower noise. The outcome is a crisp binary picture with distinct edges, which we use to extract and filter contours based on their area, adding to these outlines to see the divided road networks. Lastly, a Hough Transform is used, with settings set to capture the important linear edges that correlate to roadways, in order to recognise straight line segments. The road segments are accurately isolated from the satellite images thanks to the two-tiered method of Sobel edge detection and Hough Transform.

Listing 1: Sobel filter code

```
def Soble_fun(self):
    gray_image = cv2.cvtColor(self.image, cv2.
```

```
    ↪ COLOR_BGR2GRAY)

normalized_image = cv2.normalize(gray_image, None,
    ↪ alpha=0, beta=1, norm_type=cv2.NORM_MINMAX,
    ↪ dtype=cv2.CV_32F)

clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize
    ↪ =(8, 8))
equalized_image = clahe.apply(np.array(
    ↪ normalized_image*255, dtype=np.uint8))

# Experiment with the Gaussian blur to further
    ↪ suppress noise
blurred_image = cv2.GaussianBlur(equalized_image, (9,
    ↪  9), sigmaX=2, sigmaY=2)

# Apply Sobel filters to detect edges
sobelx = cv2.Sobel(blurred_image, cv2.CV_64F, 1, 0,
    ↪ ksize=5)  # Experiment with the kernel size
sobely = cv2.Sobel(blurred_image, cv2.CV_64F, 0, 1,
    ↪ ksize=5)

# Calculate the gradient magnitude and normalize
sobel_magnitude = np.sqrt(sobelx**2 + sobely**2)
sobel_magnitude = cv2.normalize(sobel_magnitude, None
    ↪ , 0, 255, cv2.NORM_MINMAX)
sobel_magnitude = np.uint8(sobel_magnitude)

# Dilate the edges to make them more pronounced
kernel_dilation = np.ones((2,2), np.uint8)
dilated_sobel_magnitude = cv2.dilate(sobel_magnitude,
    ↪  kernel_dilation, iterations=1)

# Apply Otsu's method to find an optimal threshold
_, otsu_thresh_sobel = cv2.threshold(
    ↪ dilated_sobel_magnitude, 0, 255, cv2.
    ↪ THRESH_BINARY + cv2.THRESH_OTSU)

# Apply morphological closing followed by opening to
    ↪ clean up the image
kernel_morph = cv2.getStructuringElement(cv2.
    ↪ MORPH_CROSS, (3,3))
closed_sobel = cv2.morphologyEx(otsu_thresh_sobel,
    ↪ cv2.MORPH_CLOSE, kernel_morph, iterations=2)
opened_sobel = cv2.morphologyEx(closed_sobel, cv2.
    ↪ MORPH_OPEN, kernel_morph, iterations=1)
```

```
contours, _ = cv2.findContours(opened_sobel, cv2.
    ↪ RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
filtered_contours = [cnt for cnt in contours if cv2.
    ↪ contourArea(cnt) > 100]  # Filter by area

contour_output = np.zeros_like(opened_sobel)
cv2.drawContours(contour_output, filtered_contours,
    ↪ -1, (255), thickness=cv2.FILLED)

kernel_maxpool = np.ones((4, 4), np.uint8)
maxpooled_image = cv2.dilate(contour_output,
    ↪ kernel_maxpool)

mask2 = maxpooled_image > maxpooled_image.mean() *
    ↪ 1.2
return mask2
```

4.2 Canny Edge Detection:

Canny Edge Detection algorithm is well-known for its accuracy in finding edges using a multi-threshold approach. This technique works well at reducing noise while maintaining edge integrity, which is important for distinguishing roads in the midst of intricate rural textures. The image is first smoothed using a Gaussian filter, and then the edges are highlighted using a gradient calculation. The final step of edge tracking via hysteresis reinforces the continuity of road edges. Non-maximum suppression and double thresholding are then used to identify strong and weak edges. The success of the Canny approach is critical to obtaining the granularity required for high-fidelity road segmentation.

Since edge detection does not require colour information, our Canny-edge Detection methodology starts with the input image being converted to grayscale. After that, the grayscale image is normalised to bring its intensity range into uniformity, which improves contrast and makes edge detection more efficient. In order to further enhance the image quality, Contrast Limited Adaptive Histogram Equalisation (CLAHE) is used, particularly in low contrast or uneven illumination areas. By regionally adjusting the contrast of the image, this approach improves its uniformity across various regions. The use of CLAHE is essential because it improves edge definition and gets the picture ready for further edge detection steps.

The image is subjected to a larger kernel size Gaussian blur after the preparation stages. By blurring, undesirable features and noise are efficiently reduced, which prevents incorrect edge detection during the Canny process. Next, to locate the edges within the blurry image, the Canny edge detector is used with carefully adjusted lower and higher thresholds. To maintain a balanced edge detection sensitivity, the higher threshold is kept at three times the lower

threshold, while the lower threshold is set to capture real edges. Following edge identification, we perform dilatation with a kernel to make the edges more visible and thicker, so that they stand out more in the following step. On these brightened edges, a Hough Line Transform is then applied to identify and distinguish linear characteristics in the picture. When it comes to removing straight line segments—which are crucial for locating road constructions in satellite imagery—this transform is especially useful. The road networks are clearly visualised in the resulting line image, which highlights the efficacy of the Canny-edge Detection technology in our project. It also displays the discovered features.

Listing 2: Canny filter code

```
def Canny(self):
    gray_image = cv2.cvtColor(self.image, cv2.
        ↪ COLOR_BGR2GRAY)

    normalized_image = cv2.normalize(gray_image, None,
        ↪ alpha=0, beta=1, norm_type=cv2.NORM_MINMAX,
        ↪ dtype=cv2.CV_32F)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize
        ↪ =(8, 8))
    equalized_image = clahe.apply(np.array(
        ↪ normalized_image*255, dtype=np.uint8))

    # Apply Gaussian blur with a larger kernel to smooth
        ↪ the image more
    blurred = cv2.GaussianBlur(equalized_image, (9, 9),
        ↪ 0)

    # Fine-tune the Canny edge thresholds again
    low_threshold = 45  # Lower this if you're missing
        ↪ real edges
    high_threshold = low_threshold * 3  # Keep the upper:
        ↪ lower ratio of about 3:1

    # Apply Canny edge detection
    edges = cv2.Canny(blurred, low_threshold,
        ↪ high_threshold)

    # Increase the thickness of the edges by dilating the
        ↪  edge image
    kernel = np.ones((4,4), np.uint8)
    brighter_edges = cv2.dilate(edges, kernel, iterations
        ↪ =1)
```

```
    return brighter_edges
```

4.3 Laplacian Edge Detection:

This algorithm combines Gaussian smoothing with the Laplacian operator to effectively localize edge detection in noisy image data. The Gaussian filter first attenuates high-frequency noise, creating a smoothed image which is then processed by the Laplacian operator to identify regions with rapid intensity changes—characteristics of edge points. This method is especially adept at revealing finer details in the road infrastructure, as it responds emphatically to edges against varied backgrounds, thereby making it an invaluable tool in our arsenal for precise road delineation in satellite imagery.

Listing 3: Code of laplacian filter

```
def Laplacian_fun(self):
    gray_image = cv2.cvtColor(self.image, cv2.
        ↪ COLOR_BGR2GRAY)

    normalized_image = cv2.normalize(gray_image, None,
        ↪ alpha=0, beta=1, norm_type=cv2.NORM_MINMAX,
        ↪ dtype=cv2.CV_32F)

    clahe = cv2.createCLAHE(clipLimit=2.0, tileGridSize
        ↪ =(8, 8))
    equalized_image = clahe.apply(np.array(
        ↪ normalized_image*255, dtype=np.uint8))

    # Experiment with the Gaussian blur to further
        ↪ suppress noise
    blurred_image = cv2.GaussianBlur(equalized_image, (9,
        ↪  9), sigmaX=2, sigmaY=2)

    # Use a bilateral filter for noise reduction while
        ↪ preserving edges
    bilateral_filtered_image = cv2.bilateralFilter(
        ↪ equalized_image, d=11, sigmaColor=90,
        ↪ sigmaSpace=90)

    # Apply the Laplacian filter using cv2.Laplacian with
        ↪  a larger kernel size
    laplacian = cv2.Laplacian(bilateral_filtered_image,
        ↪ cv2.CV_64F, ksize=5)

    # Convert back to uint8
```

```
    laplacian = np.uint8(np.absolute(laplacian))

    mask1 = laplacian > laplacian.mean() * 2
    return mask1
```

4.4 Hybrid Sobel-Canny Method:

The Hybrid Sobel-Canny Method combines the strong edge detection powers of both Sobel and Canny algorithms. This combined method takes advantage of the greater noise reduction of the Canny technique as well as the effectiveness of the Sobel operator in recording edge orientations. The roads' gradients are first highlighted using the Sobel operator, and their exact edge tracking is subsequently enhanced by the Canny algorithm. The specific shortcomings of each technique are lessened by combining the two, leading to a more precise and trustworthy detection of road boundaries. The purpose of this hybrid method is to improve the precision and lucidity of road segmentation in our dataset through a strategic fusion.

By merging the unique strengths of the Sobel and Canny algorithms, we utilise a two-pronged strategy for edge identification in the Hybrid Sobel-Canny Method. The goal of this technique is to improve the accuracy of road segmentation from satellite data by utilising the directional edge detection of the Sobel filter and the noise-resistant capabilities of the Canny algorithm. The Canny edge detector application is the initial step in the procedure. The Canny algorithm is quite good at detecting edges in an image, both strong and weak, which helps it capture the important road outlines even in places with uneven contrast. A binary mask (mask1) that highlights the edges found by the Canny method is the result of this phase.

Simultaneously, the image's intensity gradients are used to identify edges using the Sobel operator. By emphasising both vertical and horizontal edges, the Sobel technique creates an extensive edge map. Another binary mask (mask2), which represents the gradient-based edges, is the result of the Sobel edge detection process.The combination of these two masks forms the basis of our methodology. Masks 1 and 2 are combined by the 'Final fun' function via a bitwise AND operation. This combination (mask3) combines the gradient-based sensitivity of Sobel with the accurate edge detection of Canny. The result is further refined by integrating the original image's green channel intensity, which improves the contrast between the roads and the surrounding scenery. The resultant final mask (mask5) from this procedure shows a sophisticated and accurate road network recognition, proving that merging Sobel and Canny approaches works well for complex image segmentation problems.

Listing 4: Code of Hybrid Sobel-Canny filter

```
def Hybrid_Canny_Soble(self):
  mask1 = self.Canny()
```

```
mask1 = mask1 > mask1.mean() * 1.2
mask2 = self.Soble_fun()

mask3= (mask1 & mask2)

g = self.image[:,:,1]
g = g < g.mean()*1.2

mask5 = mask3 & g

return mask5
```

4.5 Hybrid Sobel-Laplacian Method:

Hybrid Sobel-Laplacian Method combines the edge localization capabilities of the Laplacian with the directional sensitivity of the Sobel operator. Using the Sobel operator to identify edge orientation and strength, this method is refined by the Laplacian's capacity to identify regions of abrupt intensity shift. Combining these two methods yields a more detailed and nuanced edge identification, which is especially useful for distinguishing the complex and frequently delicate road networks from the satellite data. By balancing the Laplacian's precision with Sobel's wide detection scope, the hybridised approach seeks to maximise the segmentation quality.

We produce greater road segmentation in satellite data by combining the unique characteristics of the Sobel and Laplacian edge detection approaches in our Hybrid Sobel-Laplacian Method. The Sobel function (Soble fun) calculates the gradient magnitude in the image's horizontal and vertical axes with the goal of retrieving edge information. The detection of edges with different orientations is made possible by this gradient-based method, which is essential for locating road networks that might not be completely aligned either vertically or horizontally. A binary mask (mask2) that emphasises the important edges found by the Sobel filter is the product of the Sobel process. This mask adds a critical layer of information for road segmentation by highlighting the image's boundaries and linear features.

The Laplacian filter, which is renowned for its capacity to record edge information based on the second derivative of the picture intensity, is simultaneously used by the Laplacian function (Laplacian fun). This technique is excellent in emphasising areas of abrupt changes in intensity, which are typical of edges in a picture. This helps to enhance the gradient information that the Sobel method captures. Another binary mask (mask1), which highlights the edges found by the Laplacian approach, is the result of the Laplacian filter. Using a bitwise AND operation, the Final-fun function merges these two masks in the last step to create a merged mask (mask3) that combines the edge detection powers of both approaches. Road segmentation becomes more accurate as a result of this integration, which improves edge detection's durability and accuracy.

Furthermore, the last action entails using a max pooling procedure to reduce the dimensionality of the combined mask while maintaining the necessary edge information, providing a more concentrated and succinct depiction of the road network in the picture.

Listing 5: Code of Hybrid Sobel-Laplacian filter

```
def Hybrid_Laplacian_Soble(self):
    mask1 = self.Laplacian_fun()
    mask2 = self.Soble_fun()

    mask3= (mask1 & mask2)

    g = self.image[:,:,1]
    g = g < g.mean()*1.2

    mask5 = mask3 & g

    return mask5
```

4.6 Hybrid Canny-laplacian Method:

The Hybrid Canny-Laplacian Method is a customised strategy created to improve road segmentation by utilising the advantages of both the Canny and Laplacian algorithms. The Canny algorithm's strong edge recognition and noise suppression capabilities are the first step in this process, which guarantees that just the roads' core structure is captured and not any extraneous details get in the way. Next, the identified edges are sharpened and the localization of road boundaries is improved by applying the Laplacian operator to the findings. This combination produces a more continuous and defined portrayal of roadways amid the intricate rural environment of our satellite images, while simultaneously improving the segmentation process and lowering the possibility of fragmented edges.

We combine the unique strengths of the Canny and Laplacian edge detection algorithms in the Hybrid Canny-Laplacian Method to improve road segmentation in satellite data. Our method starts with the Canny algorithm (canny-fun), which is well-known for its ability to detect a large number of edges while minimising noise. By using this technique, significant edge features in the image are located and highlighted in a binary mask (mask1). The Canny edge detector is very helpful for defining roadways, whose visibility may fluctuate owing to external circumstances, because it can detect both strong and weak edges.

In parallel, the Laplacian filter (Laplacian-fun) is used to highlight the image's sharp edge-characteristic intensity variations. The Canny approach achieves broader edge detection, but this filter is especially effective at emphasising finer features. Another binary mask (mask2) is created from the Laplacian

filter's output, emphasising the edges that were found using this technique. Using a bitwise AND operation, the Final-fun function combines the two masks in the last stage to create a combined mask (mask3) that takes advantage of the advantages of both Canny and Laplacian edge detection. By improving the segmentation process's durability and accuracy, this integration makes it possible to identify road networks with greater precision. Furthermore, the final algorithm includes the green channel's intensity data of the source image, improving the accuracy of the segmentation even further. The outcome shows how well Canny and Laplacian algorithms work together to efficiently recognise road networks in complicated image segmentation challenges.

Listing 6: Code of Hybrid Canny-Laplacian filter

```
def Hybrid_Laplacian_Canny(self):
    mask1 = self.Laplacian_fun()
    mask2 = self.Canny()


    g = self.image[:,:,1]
    g = g < g.mean()*1.2

    mask3= (mask1 & mask2) & ~g

    return mask3
```

4.7 Adaptive Thresholding with Canny:

we innovated an adaptive thresholding technique that builds upon the Canny edge detection algorithm, which we refer to as the Adaptive Thresholding with Canny method. The essence of this method lies in its ability to dynamically adjust the threshold values based on the intensity of the edges detected by the Canny algorithm (brighter-edges). To enhance the selectivity of the road edges, we employed a logical operation that considers the mean intensity values of the red channel (r) of our image. This operation is designed to filter out non-road edges, relying on the heuristic that roads typically have a distinct intensity profile compared to their surroundings in the red channel of the spectrum.

Further refining our segmentation process, we introduce a logical mapping (map) that intensifies the detected edges by setting a threshold at 1.3 times the mean of the brighter-edges. We also consider the green channel (g), applying a scaling factor of 0.2 to its mean intensity to discriminate against vegetative areas that could potentially be misinterpreted as roads. The final logical step combines the red channel condition with the edge map and inverses the green channel condition (l), yielding a binary image (fin). This binary image represents the final road segmentation output, where roads are delineated with heightened precision. This Adaptive Thresholding with Canny method demonstrates our project's commitment to achieving accurate road segmentation by

tailoring traditional algorithms to suit the specific challenges presented by satellite imagery.

Listing 7: Code of Adaptive thresholding of Canny filter

```
def Canny_logical(self):
  brighter_edges = self.Canny()
  r = self.image[:,:,0] > self.image[:,:,0].mean()
  map = brighter_edges>brighter_edges.mean()*1.3
  g = (self.image[:,:,1] > self.image[:,:,1].mean()*0.2)
  l = r | ~g
  fin = ((r & map) & l)
  return fin
```

5 Evaluation metrics:

In our evaluation approach, we meticulously assessed the performance of our road segmentation algorithm using a suite of metrics. After processing the image to highlight edges, we created a binary representation (binary-image) to facilitate the analysis. We compared this binary image against a binarized version of our ground truth, which was derived by applying a threshold based on the mean intensity of the ground truth image.

We resized the ground truth image to match the dimensions of our processed output, ensuring an accurate comparison using the Structural Similarity Index (SSIM). SSIM provided us with a measure of the similarity between the two images, reflecting the quality of our segmentation.

Additionally, we computed the precision, recall, F1 score, and accuracy by flattening the binary images into 1D arrays and performing a direct comparison of the pixel values. Precision gave us the proportion of true positive road identifications, recall indicated the algorithm's ability to detect all actual roads, the F1 score provided a harmonic mean of precision and recall, and accuracy reflected the overall correctness of our segmentation. These metrics collectively offered a comprehensive view of our model's performance.

Listing 8: Evaluation Metrics

```
def Decision_Metrics(self,op_put_image):

    #plt.imshow(ground_truth,cmap='gray')
    if op_put_image.max() != True:

      line_image = (op_put_image>op_put_image.mean())
    else:
      line_image = op_put_image
```

```
binary_image = np.array(line_image) * 255

# Convert to 8-bit unsigned integer
binary_image = binary_image.astype(np.uint8)


# Create a grayscale image
line_gray_image = cv2.cvtColor(binary_image, cv2.
    ↪ COLOR_GRAY2BGR)



if len(line_gray_image.shape) == 3:
    line_gray_image = cv2.cvtColor(line_gray_image,
        ↪ cv2.COLOR_BGR2GRAY)

# Resize the ground truth to match the Hough
    ↪ Transform output
# This step is crucial to compare the images using
    ↪ SSIM
ground_truth = cv2.resize(self.ground_truth, (
    ↪ line_gray_image.shape[1], line_image.shape[0]))


similarity_index, diff = ssim(self.ground_truth,
    ↪ line_image, full=True)

binarized_ground_truth = (self.ground_truth > self.
    ↪ ground_truth.mean())

# Flatten the binary images to 1D arrays for
    ↪ comparison
binarized_ground_truth_flat = binarized_ground_truth.
    ↪ flatten()
binarized_line_image_flat = line_image.flatten()


## Calculate precision, recall, F1 score, and
    ↪ accuracy
precision = precision_score(
    ↪ binarized_ground_truth_flat,
    ↪ binarized_line_image_flat)
recall = recall_score(binarized_ground_truth_flat,
    ↪ binarized_line_image_flat)
f1 = f1_score(binarized_ground_truth_flat,
    ↪ binarized_line_image_flat)
```

```
accuracy = accuracy_score(binarized_ground_truth_flat
    ↪ , binarized_line_image_flat)

## Print out the metrics
print(f'Precision: {precision:.4f}')
print(f'Recall: {recall:.4f}')
print(f'F1 Score: {f1:.4f}')

print(f'SSIM: {similarity_index}')
```

6 Contributions by Team Members:

Image	Method	SSIM Score
L97c	Sobel with hough tranform	0.682
L97c	canny with hough tranform	0.666
L97c	Laplacian with hough tranform	0.688
L97c	Hybrid Sobel_Canny method	0.665
L97c	Hybrid Laplacian_Sobel method	0.666
L97c	Hybrid Laplacian_Canny method	0.660
L97c	Adaptive thresholding with canny method	0.668
W107d	Sobel with hough tranform	0.779
W107d	canny with hough tranform	0.777
W107d	Laplacian with hough tranform	0.792
W107d	Hybrid Sobel_Canny method	0.774
W107d	Hybrid Laplacian_Sobel method	0.774
W107d	Hybrid Laplacian_Canny method	0.773
W107d	Adaptive thresholding with canny method	0.781
L97d	Sobel with hough tranform	0.828
L97d	canny with hough tranform	0.831
L97d	Laplacian with hough tranform	0.826
L97d	Hybrid Sobel_Canny method	0.831
L97d	Hybrid Laplacian_Sobel method	0.833
L97d	Hybrid Laplacian_Canny method	0.835
L97d	Adaptive thresholding with canny method	0.833
L96b	Sobel with hough tranform	0.879
L96b	canny with hough tranform	0.777
L96b	Laplacian with hough tranform	0.792
L96b	Hybrid Sobel_Canny method	0.874
L96b	Hybrid Laplacian_Sobel method	0.774
L96b	Hybrid Laplacian_Canny method	0.873
L96b	Adaptive thresholding with canny method	0.681

7 Contributions by Team Members:

Coordinated project workflow and algorithm integration through online meetings and study sessions at the university's technical library.

Advanced the development of the Gradient-based Sobel Operator for edge detection, meticulously fine-tuned by Shanmukha Datta and Prasaad Reddy.

Expert application of the Canny Edge Detection algorithm to isolate potential road networks by Prasaad Reddy and Izzaz Ali.

Iterative processing and enhancement of image sets for optimal edge recognition spearheaded by Shanmukha Datta.

Analysis and debugging of edge detection outputs from the Laplacian of Gaussian method, jointly tackled by Izzaz Ali and Shanmukha Datta.

Innovated on the Hybrid Sobel-Laplacian Method for precise edge mapping, led by Prasaad Reddy.

Execution of the final ensemble approach, synergizing Sobel, Canny, and Laplacian filters for robust segmentation, by Izzaz Ali.

Generation and verification of Ground Truth datasets for method validation, a joint effort by Shanmukha Datta and Izzaz Ali.

Thorough documentation of each stage of the project, from initial concept to final analysis, collectively compiled by all team members.

8 Timeline:

Updates	Dates	Status
Initial Team Meeting and Role Assignment	07-Sept-2023	
Development of Methodology Framework	14-Sept-2023	Completed
Implementation of Sobel-edge Detection	23-Sept-2023	Completed
Implementation of Canny-edge Detection	30-Sept-2023	Completed
First Integration of Hybrid Methods	05-Oct-2023	Completed
Revision and Optimization of Algorithms	20-Oct-2023	Completed
Validation of Edge Detection Accuracy	02-Nov-2023	Completed
Performance Analysis and Updates	15-Nov-2023	Completed
Final Review and Project Completion	02-Dec-2023	Completed
Compilation of Final Report and Slides	04-Dec-2023	Completed