AUTOMATIC ROAD EXTRACTION FROM MULTISPECTRAL HIGH RESOLUTION SATELLITE IMAGES

1.Abstract:

In this project I have extracted the road from a high-resolution satellite image with the help of computer vision technique that uses the edge detection process and contour making process. The contour generated is stacked over the original image to display the result. This is the simplest approach in comparison to the SVM, Fuzzy classification and other approaches. The output is easy to understand and computation is easy and less time-consuming.

1. Introduction:

The road detection algorithm uses computer vision techniques to identify road borders in satellite images. The OpenCV (cv2) library is used to process images in the algorithm's Python implementation, while NumPy is used to do numerical computations. The algorithm's primary objective is to locate road sections in satellite photos, which it then uses to create a binary image with road contours superimposed on the background image. The work deals with extraction of roads from satellite images is a challenging domain compared to extraction from aerial images as satellite images are noisy and of lower resolution.

2. Methodology:

The following steps are performed in the road detection algorithm:

Load the satellite image: The satellite image is loaded using cv2.imread() function.(fig:1)

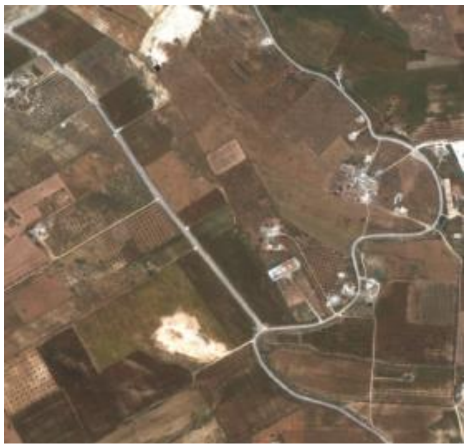

Fig:1

Convert to grayscale: The loaded image is converted to grayscale using cv2.cvtColor() function to simplify the image and reduce noise. This grayscale conversion helps simplify the image and reduce noise, as it removes color information and keeps only the intensity values. (fig:2)

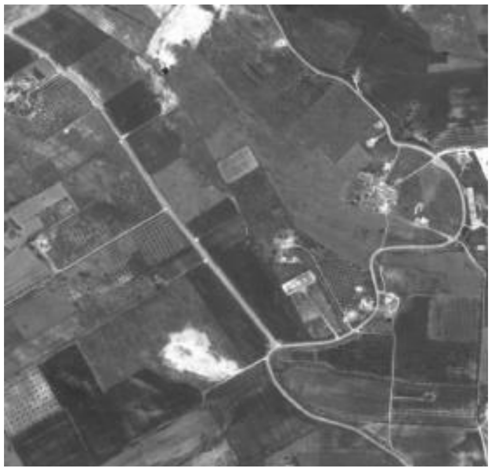

Fig:2

Gaussian blur: A Gaussian blur is applied to the grayscale image using the cv2.GaussianBlur() function. Gaussian blur is a type of low-pass filter that smoothens the image by averaging the pixel intensities in a local neighborhood(fig:3). This step helps to reduce noise in the image and create a smoother grayscale image, which can enhance the performance of the edge detection. We are using a gaussian blur matrix of size (7,7).

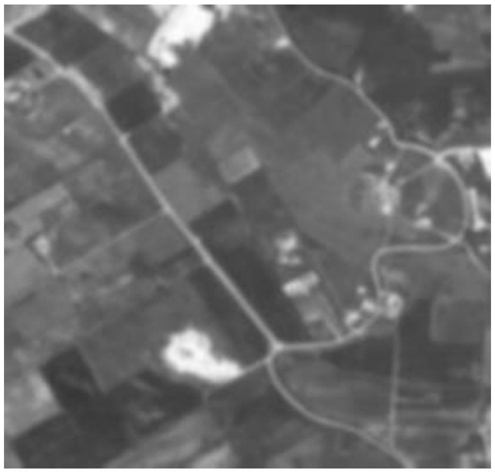

Fig:3

Edge detection: Canny edge detection is performed using the cv2.Canny() function. The Canny edge detector is a popular edge detection algorithm that detects edges based on intensity gradients. It identifies areas of rapid change in intensity, which typically correspond to edges in an image. The resulting edge map is a binary image with white pixels representing edges and black pixels representing non-edges.

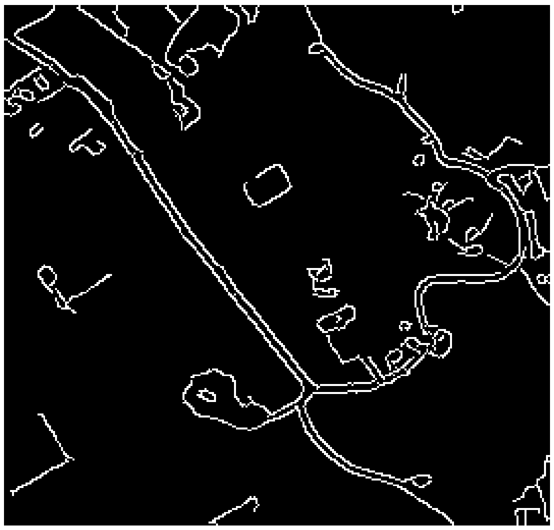

Fig:4

Thresholding: The resulting edge map is threshold using the cv2.threshold() function. Thresholding is a process of converting a grayscale image into a binary image by assigning a threshold value. Pixels with intensity values above the threshold are set to white (255), while pixels with intensity values below the threshold are set to black (0). This step helps to create a binary image with clear road edges, where the road edges are represented by white pixels and the non-road areas are represented by black pixels.

Fig:5

Morphological closing: Morphological closing operation is applied using the cv2.morphologyEx() function. Morphological closing is a dilation operation followed by an erosion operation, used to fill in gaps in the binary image and make the edges continuous. It helps to smoothen the road contours and close small gaps between road edges that may have been caused by noise or thin road segments.

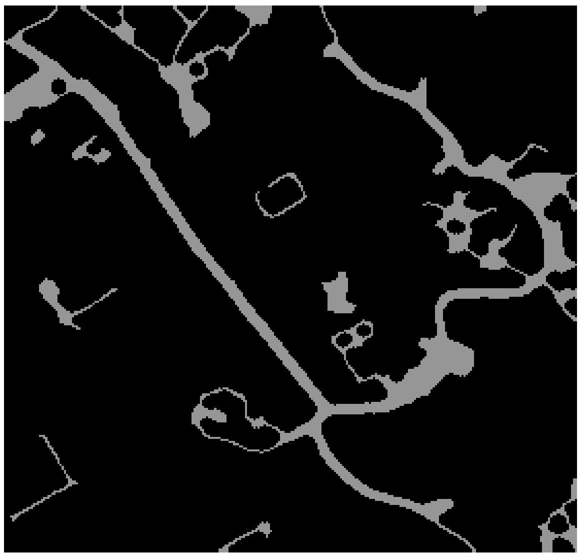

Fig:6

Thinning: The cv2.ximgproc.thinning() function is used to carry out the thinning procedure. Thinning, also known as skeletonization, is a morphological procedure that maintains connection while thinning edges to a single pixel width. To obtain a thin representation of the road network, this stage aids in further refining the road outlines.

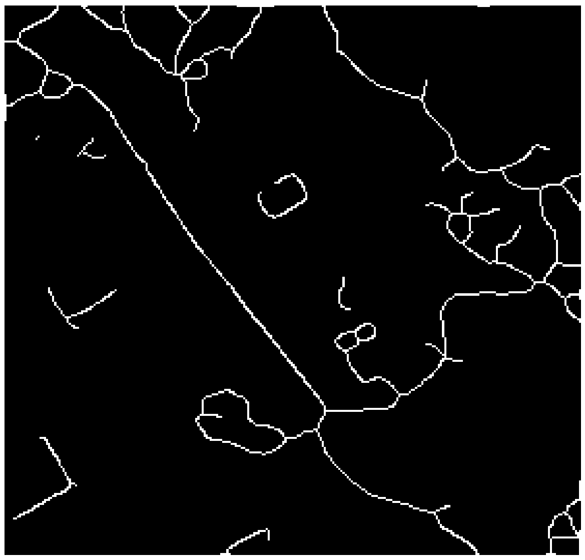

Fig:7

Contour Making:

In this code the contour making involves three steps

1. Contour detection: Contours are detected in the thinned image using cv2.findContours() function.
2. Contour filtering: Contours with small area (less than 1000 pixels) are filtered out using contour area thresholding to remove noise.
3. Draw contours: Filtered contours are drawn on the original satellite image using cv2.drawContours() function to visualize the road contours.

In the end a separate provision is applied for contours which are not connecting to the main contour line thus mostly does not signifies road.

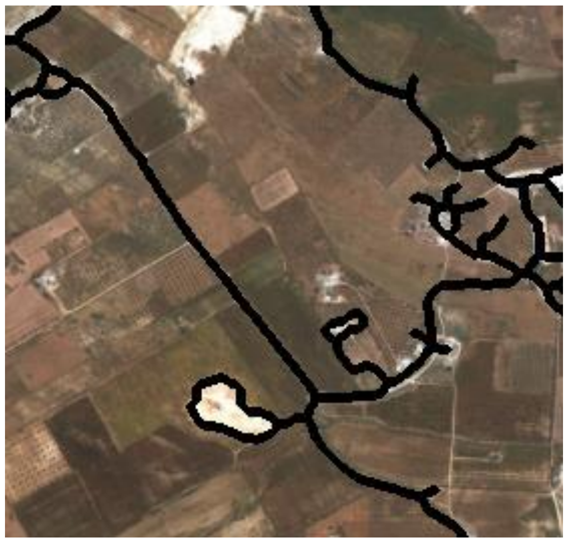

Fig:8

3. Results:

The road detection algorithm successfully detects road edges and generates a binary image with road contours on the original satellite image. The resulting road contours are visually displayed on the original image using cv2_imshow() function.

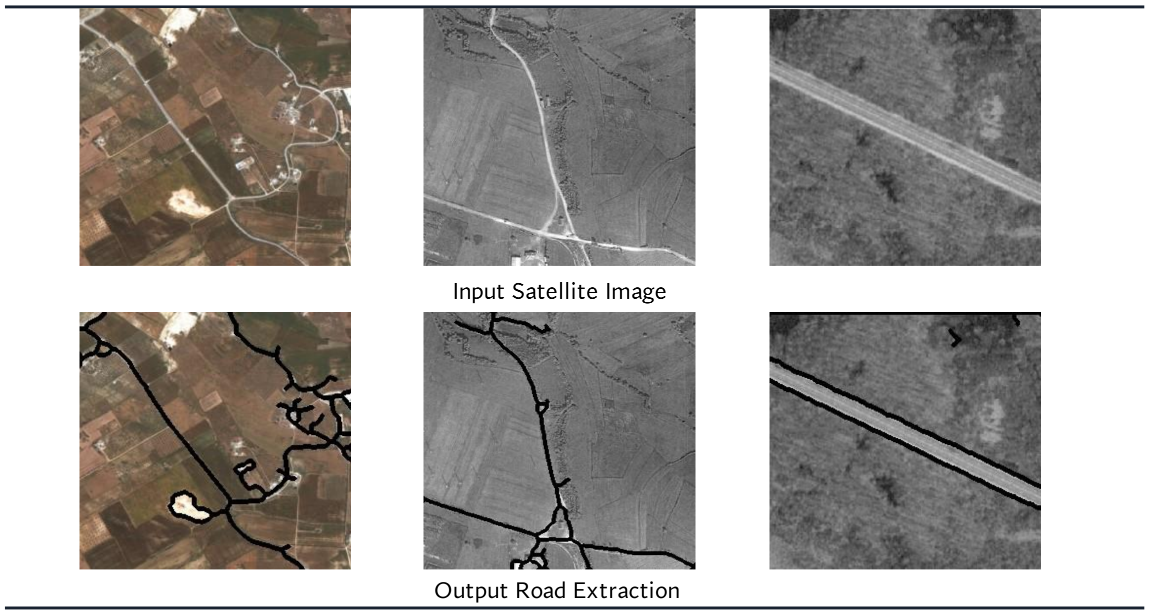

Fig:9

In the last image the double line for a single road is formed because of the large road width(fig:9).

4. Discussion:

The performance of the road detection algorithm may vary depending on the width of the roads in the satellite image. For wider roads, the algorithm may require parameter adjustments such as larger blur kernel size, higher Canny edge detection thresholds, larger morphological closing kernel, and higher contour area threshold to accurately detect the entire road width. The method used for road detection is computer vision technique that simply uses the edge detection process in creating the contour with the help of binary classification of road and non-road element.

5. Future research:

In this code have used high resolution satellite images but they are not multispectral. We can use this same code to find the road contour and then compare these different contours and select the best counter. We can also use the Machine Learning concept to classify the road elements in image.

5. Conclusion:

In conclusion, the road detection algorithm implemented in this code provides a basic framework for detecting road edges from satellite images using computer vision techniques. Further optimization may be required for accurate detection of wider road widths in specific satellite images. This algorithm can be further improved and integrated into larger systems for applications such as road network mapping, traffic analysis, and urban planning.

References:

1. Automated Road Extraction From High Resolution Satellite Images Jose Hormese , Dr. C. Saravanan.
2. AUTOMATIC ROAD EXTRACTION FROM MULTISPECTRAL HIGH RESOLUTION SATELLITE IMAGES Uwe Bacher and Helmut Mayer.
3. Automatic Road Extraction from Aerial Image John C. Trinder, Yandong Wang.
4. https://bhuvan-app1.nrsc.gov.in/globe/2d.php# "For downloading the image data."
5. https://earthexplorer.usgs.gov/ "For downloading the image data."