

Comparative analysis of edge detection image processing techniques for efficient traffic signal management system

Niranjana Chandrasekara Bharathy, Manju Shivanna Dasappanavar,

Anne Gowda Aleri Byre Gowda, Jijesh Jisha Janardhanan

Department of Electronics and Communication Engineering, Sri Venkateshwara College of Engineering, Bangalore, India

Article Info

Article history:

Received Jul 2, 2024

Revised Oct 4, 2024

Accepted Oct 7, 2024

Keywords:

Canny edge detection

Computer vision library

Image processing

Image segmentation

Machine learning

Open source

ABSTRACT

Traffic monitoring and control pose significant challenges, particularly in metropolitan areas worldwide. Traditional methods such as timers, traffic police control, and sensor-based systems have become increasingly ineffective due to escalating traffic volumes. Image processing emerges as a promising technology for enhancing traffic control systems. This research aims to address the inefficiencies of current traffic management systems (TMS) by conducting a detailed comparative analysis of conventional image processing techniques, focusing on edge-based methods. Utilizing the open source computer vision (OpenCV) Library, we evaluated various edge detection techniques based on qualitative parameters like environmental lighting, object detection, and size, as well as quantitative parameters such as average traffic density, sharpness ratio (SR), abruptness, sensitivity factors (SF), processing time, frame processing time, and average frames per second (FPS). The study finds that the Canny edge detection technique outperforms others, with an average traffic density of 30.12 across 10 frames, superior SR, and minimal processing time of 99 seconds. This makes it highly effective for traffic control applications by generating high-quality edge maps with minimal noise. Our findings suggest that improving conventional image processing techniques and integrating deep learning algorithms can further enhance TMS, leading to more efficient urban mobility and reduced environmental impact.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Niranjana Chandrasekara Bharathy

Department of Electronics and Communication Engineering, Sri Venkateshwara College of Engineering
Bangalore, Karnataka

Email: niranjana3194@gmail.com

1. INTRODUCTION

Metro cities all over the globe are suffering from severe traffic issues. The traffic jams and traffic congestion result in increased travelling time for short distances, excess fuel consumption, infrastructure strain, negative impact on life quality, negative impact on the economy, and increased air pollution. In addition to that increased accidents and frustration among commuters are the common impacts of traffic issues. The increase in traffic jams and congestion leads reduction in the efficiency of the transportation system. Nowadays, traffic management systems (TMS) [1] include traffic monitoring and data collection, traffic analysis and prediction, traffic control and regulation, public information and communication, smart infrastructure and technology, and continuous monitoring and evaluation. TMS majorly collect real-time data like traffic density, speed, congestion level, type of vehicle [2], and weather conditions and based on convenient techniques like image processing and machine learning approaches, traffic patterns and trends will be identified. The control measures are taken based on the pattern and trends. In the last stage,

a continuous monitoring and evaluation process is carried out for the optimization of TMS. Image processing [3] is the most popular technique adopted in TMS due to enhanced visual perception, quantitative analysis, monitoring and remote sensing, and creative manipulation. The significance of image processing is enabling monitoring, optimisation of traffic flow, data collection and analysis [4]. The important image processing techniques used in traffic signal management are presented in Figure 1.

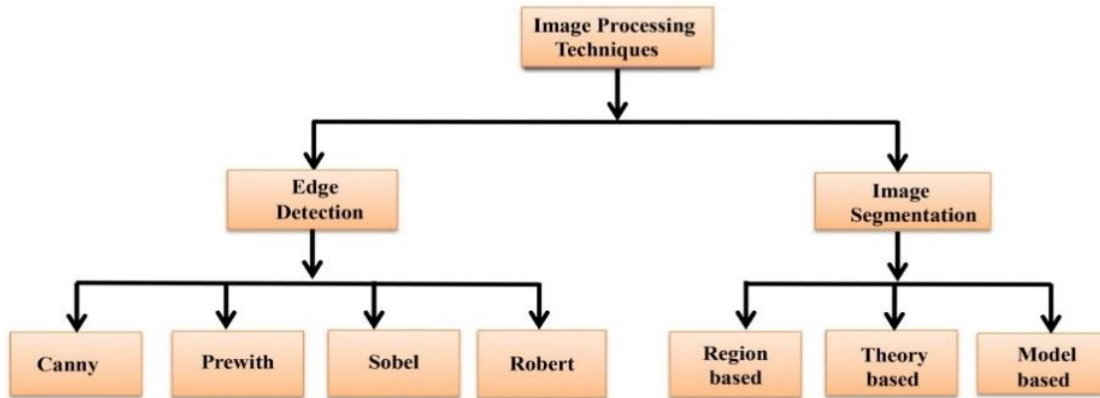


Figure 1. Classification of image processing techniques for traffic signal management

Image processing techniques have become an integral part of traffic management. They offer benefits like real-time monitoring, analysis, and decision-making capabilities at different levels of the transportation network. The block diagram of image processing in traffic signalling is shown in Figure 2.



Figure 2. Image processing in traffic signalling system

In a traffic signal management system illustrated in Figure 2, image processing plays a crucial role in ensuring efficient traffic flow and safety. Cameras installed at intersections capture real-time images of traffic conditions. The captured images are segmented to identify different elements such as vehicles, pedestrians, and background. Using techniques like object detection, vehicles and other relevant objects are detected within the segmented images. Once detected, the system recognizes the type of vehicles (car, truck, and bicycle) and the presence of pedestrians. The system compares the current image data with data from previous instances to identify changes in traffic flow, vehicle density, or other relevant parameters. To compare the techniques on the OpenCV platform, Various performance parameters can be taken into account. The performance parameters such as accuracy, traffic density, speed, robustness, edge localization, memory usage, colour, size of objects and easiness of implementation [5] are taken into consideration. By conducting such comparisons, one can determine the strengths and weaknesses of each technique and choose the most suitable one according to the application's specific requirements.

2. LITERATURE SURVEY

Image processing is one of the most promising technologies typically for the applications like TMS. In order to analyse the suitability of various image processing techniques, an extensive literature review is carried out. The pros and cons of each method are discussed using various literatures.

2.1. Conventional traffic signal management systems

Rafter *et al.* [6] explains the fact on conventional traffic signal systems have long been the cornerstone of urban traffic management, offering a systematic approach to regulating vehicular and pedestrian flow at intersections. Typically, these systems rely on pre-programmed timing sequences that dictate when each direction of traffic receives a green light, yielding to the next phase of movement. Coordinated with sensors or timers, these signals ensure efficient traffic progression while also prioritizing safety. Reducing the potential for accidents and congestion, conventional traffic signal systems are often augmented with additional features.

Jiang and Luo [7] discuss about the fact that conventional traffic signal systems face challenges in dynamically adapting to changing traffic patterns and optimizing efficiency in real-time. Fixed timing sequences may not adequately accommodate fluctuations in traffic volume, leading to inefficiencies and potential congestion during peak hours. Additionally, these systems can struggle to prioritize alternative modes of transportation, such as bicycles or public transit, which are increasingly integral to sustainable urban mobility initiatives. As urban centers evolve, there is a growing need for traffic management solutions that embrace technological advancements to enhance adaptability, responsiveness, and inclusivity within transportation networks.

According to Kumar and Singh [8] one notable disadvantage of conventional TMS lies in their limited adaptability to dynamic traffic conditions. Fixed-timing signal systems, for instance, operate on predetermined schedules that may not effectively respond to fluctuations in traffic volume throughout the day. This rigidity can lead to inefficiencies and congestion in areas with unpredictable traffic patterns. Moreover, conventional systems often prioritize vehicular traffic over alternative modes of transportation, such as bicycles or public transit, neglecting the diverse needs of urban commuters. As cities evolve and transportation trends shift towards sustainability and multi-modal solutions, there's a pressing need for TMS that offer greater flexibility, responsiveness, and inclusivity.

2.2. Image processing using edge detection

Ang *et al.* [9] highlighted that edge detection is essential for various computer vision applications, including object detection, image segmentation, and feature extraction. The primary objective of edge detection is to locate significant transitions or discontinuities in pixel intensity, which often correspond to the edges of objects or changes in texture within the image. According to Lau *et al.* [10] there are several algorithms and methods employed for edge detection, each with its strengths and limitations. The most widely used technique is the gradient-based approach, where gradients in pixel intensity are computed using operators like the Sobel, Prewitt, or Roberts operators. These operators analyze the intensity variation in the neighborhood of each pixel to identify regions of rapid change, indicating potential edges. Wu *et al.* [11] recommended that Edge detection in image processing offers several advantages, including its ability to extract important features such as object boundaries and contours, which are essentials for tasks like object recognition, image segmentation, and shape analysis. By highlighting areas of rapid intensity change, edge detection algorithms can effectively reduce the complexity of images while preserving essential information, leading to more efficient processing and analysis. Furthermore, edge detection facilitates the identification of key structures within images, aiding in the interpretation and understanding of visual data in various fields such as medical imaging, remote sensing, and robotics.

Mosavi *et al.* [12] discusses about the pros and cons of edge detection technique in details. One significant disadvantage is its sensitivity to noise, which can lead to the detection of false edges and inaccuracies, especially in images with low signal-to-noise ratios. Edge detection algorithms may struggle with variations in lighting conditions, as well as with the presence of texture or patterns that can interfere with edge identification. The choice of parameters and algorithms for edge detection can significantly impact the results, requiring careful tuning and experimentation to achieve optimal performance across different types of images and applications. Edge detection remains a valuable tool in image processing, offering indispensable insights into the structure and content of visual data. This method effectively reduces noise while accurately detecting edges with high localization and minimal false positives. However, edge detection algorithms must contend with challenges such as noise, varying lighting conditions, and the presence of weak or spurious edges, which can impact their reliability and robustness in practical applications. Nonetheless, with ongoing advancements in computer vision and image processing, edge detection remains a critical tool for extracting meaningful information and powering a wide range of applications across diverse domains [13].

2.3. Image processing using segmentation

Sadowski *et al.* [14] stated that image processing using segmentation is vital for a range of applications, including object detection, medical imaging, and satellite image analysis. By dividing an image into meaningful segments, segmentation enables the extraction of important features and the identification of

objects of interest with greater accuracy and efficiency. This process allows for better understanding and interpretation of visual data, facilitating tasks like image recognition, measurement, and classification. Saura *et al.* [15] mentioned that Segmentation methods can be broadly divided as two main types: threshold-based and region-based. Threshold-based segmentation involves setting thresholds on pixel intensity or color to separate objects from the background. This method is simple and effective for images with well-defined boundaries and uniform background. On the other hand, region-based segmentation techniques group pixels together based on similarities in their attributes, such as color or texture, using algorithms like region growing or split-and-merge. These methods are more versatile and robust, capable of handling complex images with varying intensity levels and textures. According to Nallaperuma *et al.* [16] image processing using segmentation also presents certain challenges, such as the selection of appropriate segmentation algorithms and parameters, which can significantly impact the results. Moreover, segmentation may be influenced by factors like noise, illumination variations, and object occlusions, leading to inaccuracies or incomplete segmentation. Despite these challenges, segmentation remains a powerful tool in image processing, offering valuable insights and enabling a wide range of applications across diverse fields.

2.4. Other methods

Schulz *et al.* [17] suggested that machine learning plays a pivotal role in image processing by providing powerful tools and algorithms in sake of tasks such as object detection, classification, segmentation, and enhancement. Through the utilization of large datasets and advanced algorithms, machine learning models can acquire intricate patterns and relationships within images, enabling them to make accurate predictions and decisions. In image classification, for example, convolutional neural networks (CNNs) [18] are commonly employed to automatically identify and categorize objects within images based on their features. Similarly, in object detection, models like faster R-CNN and YOLO utilize machine learning techniques to localize and classify objects of interest within images with remarkable precision and speed. Moreover, machine learning algorithms enable sophisticated image enhancement techniques, such as super-resolution and denoising, by learning from examples to generate high-quality images from low-resolution or noisy inputs. Yu *et al.* [19] used Q-learning and fast gradient-descent to figure out the best action for signal settings. They showed how well this method works with simulations. They aimed to cut down waiting time by deciding on signal timing using a linear regression algorithm. Predicting traffic volume accurately is key for scheduling traffic signals. To tackle this, another group suggested using a combination of machine learning models to forecast traffic on incoming roads.

Huang *et al.* [20] suggested an adaptive traffic signal controller using linear regression to predict green light duration for each phase, reducing vehicle queues during red periods. Another research proposes a dynamic signal timing strategy based on pedestrian volume, using radar data to minimize waiting delays. They use support vector machines to estimate queuing delays. A new traffic signal switching system is introduced, integrating object detection through machine learning. An evolutionary algorithm optimizes signal strategies to cut down on pedestrian and vehicle wait times. Xu *et al.* [21] proposed a new way to gather important traffic data like vehicle volume and red-light duration at signalized highways. They use a k-nearest neighbour classification algorithm and image processing. This data helps improve signal settings. Other researchers in [22], [23] use fuzzy logic, artificial neural networks, and genetic algorithms to optimize traffic signals. Worldwide, scientists are continuously improving signal operations at intersections to make traffic controllers smarter. Machine learning is heavily used for this purpose. Recent studies in [24], [25] employ reinforcement learning to create adaptable and smart traffic signal controllers.

3. MATERIALS AND METHODS

Image processing is vital for traffic signal management in cities. Cameras capture images at intersections to detect vehicles and pedestrians, and refine signal timings to alleviate congestion and enhance traffic flow. This technology is crucial for improving traffic efficiency and safety on busy urban roads. In order to analyse the performance of conventional techniques, popular algorithms come under these methods are analysed and tested using OpenCV environment.

3.1. Edge detection techniques

Edge detection technique in image processing is a method used to identify boundaries within an image, highlighting abrupt changes in intensity. This helps in extracting important features like edges or outlines, facilitating further analysis and processing. Edge detection in image processing is essential for identifying boundaries within an image, which highlights abrupt changes in intensity. This facilitates the extraction of crucial features such as edges or outlines, aiding further analysis and processing.

3.1.1. Canny edge detection

Canny edge detection is a widely used image processing technique known for its high accuracy and low error rates in edge detection. The method involves several steps, including noise reduction, gradient calculation, non-maximum suppression, and edge tracking by hysteresis, which together ensure precise edge detection suitable for applications such as object recognition and image segmentation. The flowchart of the Canny edge detection technique is illustrated in Figure 3. The algorithm begins by taking an image as input and proceeds through these stages to effectively identify edges while minimizing noise. This image is then converted to grayscale, simplifying the processing. Next, a Gaussian blur is applied to the grayscale image to reduce any noise present in the image. The Canny edge detection algorithm is renowned for its high accuracy and low error rates. It involves the following steps, each contributing to its effectiveness in various applications like object recognition and image segmentation:

- a) Input image: start by loading an image and converting it to grayscale for simplification.
- b) Noise reduction: apply a Gaussian blur to the grayscale image to reduce noise.
- c) Gradient calculation: compute the calculation at each pixel, highlighting areas with rapid intensity changes.
- d) Non-maximum suppression: identify local maxima in the gradient direction and suppress non-maximum pixels, preserving thin edges.
- e) Thresholding: define high and low thresholds to categorize edges as strong or weak.
- f) Edge tracking by hysteresis: retain weak edges ensuring continuous edge formation.
- g) Output: the final image is generated, where edges are highlighted against a black background.

After that, the algorithm computes the gradient magnitude and direction at each pixel. This helps to highlight areas with rapid intensity changes, which typically correspond to edges. Following the gradient calculation, non-maximum suppression is performed. This step identifies local maxima in the gradient direction and suppresses non-maximum pixels, ensuring that only thin edges are preserved. Then, high and low threshold values are defined to determine which edges are strong and which are weak. In the edge tracking by hysteresis step, strong edges above the high threshold are marked, and weak edges connected to strong edges are retained.

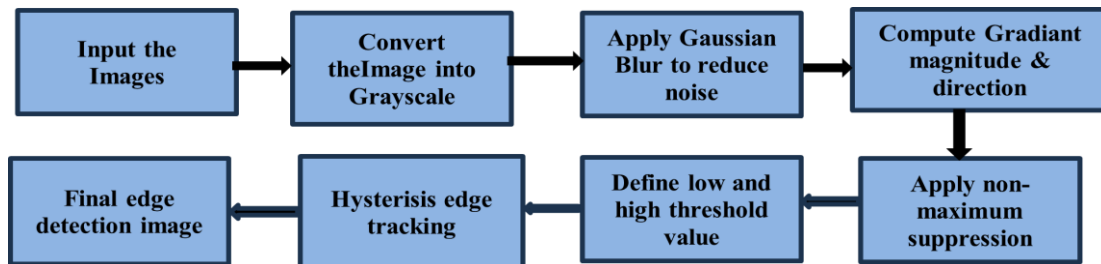


Figure 3. Canny edge detection techniques

This helps in connecting discontinuous edges and forming complete edges. Finally, the algorithm outputs the final edge-detected image, where edges are highlighted against a black background. Overall, the Canny edge detection algorithm provides a robust method for accurately detecting edges in images, making it useful in various applications such as object detection and image segmentation. The gradient magnitude is computed using (1).

$$G = \sqrt{G_x^2 + G_y^2} \tag{1}$$

In the (1) G_x is the gradient in x direction, where G_y is the gradient in y direction. Also, gradient direction is estimated using the (2).

$$\theta = \arctan\left(\frac{G_y}{G_x}\right) \tag{2}$$

3.1.2. Prewitt edge detection

Prewitt edge detection is a widely used image processing technique that applies two 3×3 kernels to compute the gradient magnitude in both the x and y directions. This helps to highlight edges in the image, making it effective for identifying sharp changes in intensity as illustrated in the Figure 4.

- Input image: convert the image to grayscale.
- Gradient calculation: apply both horizontal and vertical Prewitt kernels to calculate gradients.
- Thresholding: set a threshold to create a binary image, enhancing edges.
- Output: generate the edge-detected image.

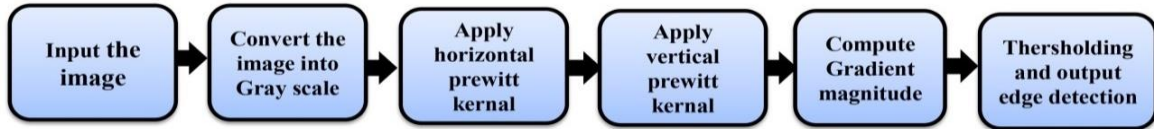


Figure 4. Prewitt edge detection techniques

To perform Prewitt edge detection, we convert the image to grayscale, apply both horizontal and vertical Prewitt kernels to calculate gradients, set a threshold to create a binary image, and output the edge-detected image. The horizontal gradient (G_x) is given in the (3):

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{pmatrix} \quad (3)$$

the vertical gradient (G_y) is shown in the (4).

$$G_y = \begin{pmatrix} -1 & -1 & 1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{pmatrix} \quad (4)$$

The flowchart shown in Figure 4 the steps involved in Prewitt edge detection, from image input to obtaining the final edge-detected result. Prewitt edge detection is a simpler technique than Canny edge detection. Prewitt involves convolving the image with two fixed kernels. Canny uses sequential operations like Gaussian blurring, gradient computation, non-maximum suppression, and hysteresis thresholding. Canny provides more accurate edge detection with reduced noise and better boundary identification.

3.1.3. Sobel edge detection

Sobel edge detection is a technique used to detect edges in images by computing the gradient approximation. It involves convolving the image with a pair of 3×3 kernels to calculate the gradients in the horizontal and vertical directions, which are then combined to determine the magnitude and orientation of edges. This method is simple, effective, and widely used in object detection and image segmentation. The flow chart of Sobel edge detection is presented in Figure 5.

- Input image: convert the image to grayscale.
- Horizontal gradient: apply a horizontal Sobel kernel to detect horizontal gradients.
- Vertical gradient: apply a vertical Sobel kernel to detect vertical gradients.
- Gradient magnitude: compute the gradient magnitudes.
- Thresholding: define a threshold to enhance edges.
- Output: produce the edge-detected image.

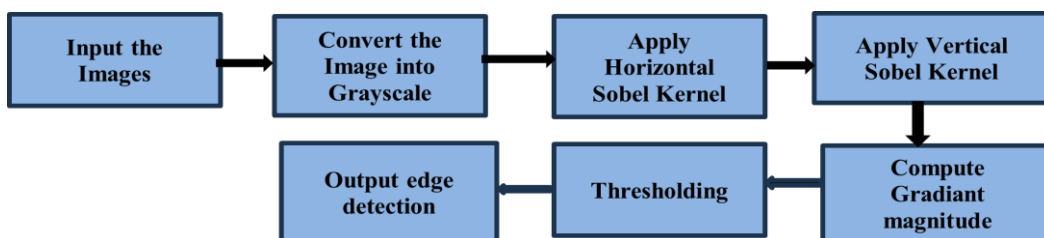


Figure 5. Sobel edge detection technique

Begin by inputting an image and converting it to grayscale to facilitate processing. Apply a horizontal Sobel kernel through convolution to detect horizontal gradients, followed by a vertical Sobel kernel to identify vertical gradients. Compute the gradient magnitudes for each pixel and establish a threshold value. Set pixels with gradient magnitudes above the threshold to white and those below to black. Conclude by outputting the edge-detected image. This process enhances edges in the image, making them stand out against the background. The horizontal and vertical sobel kernel is shown in the (5) and (6) respectively.

$$G_x = \begin{pmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{pmatrix} \tag{6}$$

$$G_y = \begin{pmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{pmatrix} \tag{7}$$

Sobel edge detection is a simpler technique that calculates gradients using fixed Sobel kernels in the horizontal and vertical directions. It’s computationally efficient but can be sensitive to noise, resulting in thicker edges. Conversely, Canny edge detection is more sophisticated, employing multiple steps such as Gaussian blurring, gradient computation, non-maximum suppression, and hysteresis thresholding. Canny edge detection produces thinner and more accurate edges, making it suitable for applications where precise edge localization and noise tolerance are crucial, albeit with a higher computational cost. In contrast, Sobel edge detection is faster and more straightforward, making it suitable for simpler edge detection tasks.

3.1.4. Robert edge detection

Robert edge detection is an image processing technique that identifies edges by applying a specific kernel to the image. While similar to Sobel edge detection, it utilizes a distinct kernel. The Robert edge detection method employs two 2×2 matrices, which are convolved with the image to compute horizontal and vertical gradients. These gradients are then combined to detect edges in the image. Due to its simplicity and computational efficiency, Robert edge detection is well-suited for real-time edge detection applications. The flow chart of Robert edge detection technique is shown in Figure 6. The expression for horizontal Robert kernel and vertical Robert kernel is shown in the (8) and (9) respectively.

$$G_x = \begin{pmatrix} 1 & 0 \\ 0 & -1 \end{pmatrix} \tag{8}$$

$$G_y = \begin{pmatrix} 0 & 1 \\ -1 & 0 \end{pmatrix} \tag{9}$$

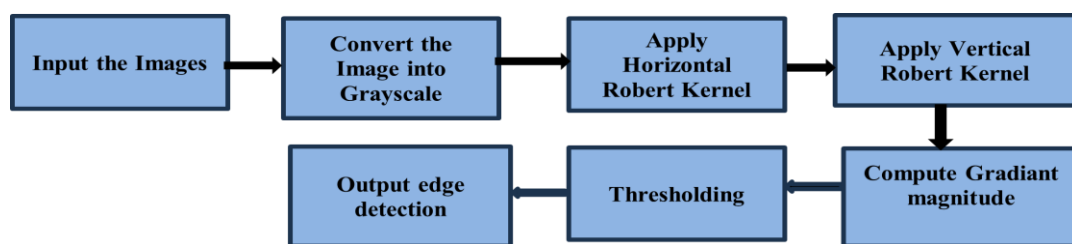


Figure 6. Robert edge detection technique

Edge detection techniques such as Robert, Sobel, and Prewitt are used to detect edges in an image. Robert and other basic techniques work by convolving the image with specific kernels to detect edges, which makes them computationally simpler and faster. They may produce thicker edges and be sensitive to noise. Alternatively, the Canny edge detection algorithm represents an advanced approach encompassing several phases: Gaussian smoothing, gradient computation, suppression of non-maxima, and threshold determination via hysteresis. It provides superior edge detection performance by producing thinner and more precise edges, making it suitable for applications where accurate edge localization and noise reduction are crucial. However, it requires more computational resources. In summary, while basic techniques like Robert, Sobel, and Prewitt are faster and simpler, Canny provides better performance at the cost of increased complexity and computational

requirements. This process enhances edges in the image, making them stand out against the background. Robert edge detection uses 2×2 kernels to calculate gradients:

- a) Input image: convert the image to grayscale.
- b) Gradient calculation: apply horizontal and vertical Robert kernels to calculate gradients.
- c) Thresholding: combine gradients to identify edges.
- d) Output: generate the edge-detected image.
- e) The flowchart for robert edge detection is shown in Figure 6

3.2. Implementation of Image processing technique in openCV environment

In order to compare popular edge detection techniques and image segmentation techniques, each algorithms are developed in openCV platform using python programming. The steps of implementation are described here. Edges that are connected to strong edges with a magnitude above threshold1 are also considered valid edges. Finally, display both the original and the edge-detected image using `cv2.imshow` and then, pause for user input to terminate the windows using `cv2.waitKey` and `cv2.destroyAllWindows`.

The algorithm for implementing the edge detection technique in openCV platform is briefly illustrated here. Consider the steps for the implementation:

- a) Import OpenCV: import the OpenCV library (`cv2`).
- b) Load image: acquire the input image via `cv2.imread`.
- c) Grayscale conversion: transform the image to grayscale utilizing `cv2.cvtColor`.
- d) Noise reduction: implement Gaussian blur through `cv2.GaussianBlur` to mitigate noise.
- e) Edge detection: implement the chosen edge detection method using the respective OpenCV functions.
- f) Thresholding: for Canny, set Threshold1 (lower threshold) and Threshold2 (upper threshold).
- g) Display results: display both the original and edge-detected images using `cv2.imshow`. Use `cv2.waitKey` and `cv2.destroyAllWindows` to manage window display.

3.3. Experimental setup

The experimental setup for our research on edge detection techniques in traffic signal management involved multiple stages, encompassing various methodologies, instruments, materials, procedures, and measurements. The primary instrument used for capturing traffic images was a high-resolution camera, strategically positioned at multiple intersections to capture clear images of vehicles and pedestrians. These images were then transformed using the OpenCV in a Python programming environment. The edge detection techniques evaluated included Canny, Prewitt, Sobel, and Robert methods. Each edge detection technique was implemented by procuring the OpenCV library and stocking the traffic images employing `cv2.imread`. Images were optionally converted to grayscale using `cv2.cvtColor` to simplify processing. Gaussian blur (`cv2.GaussianBlur`) was applied to reduce noise, with the kernel size adjusted based on the image properties. The core edge detection was performed using the respective OpenCV functions for each technique, such as `cv2.Canny` for Canny edge detection, where two threshold values were specified to differentiate between strong and weak edges. The output images were displayed using `cv2.imshow`, and user inputs were handled with `cv2.waitKey` and `cv2.destroyAllWindows` for closing windows. Each technique's performance was evaluated based on several criteria, including object size detection, environmental lighting conditions, multiple detection capabilities, and processing time. The Camera specifications is Hikvision DS-2CD2085FWD-I model, 8 MP (3840×2160) Resolution and 30 frames per second (FPS) is the frame rate. The Cameras were strategically placed at various intersections to cover different lanes of traffic, ensuring a comprehensive view of vehicle and pedestrian movements. The orientation was adjusted to capture the maximum field of view without obstructions, typically mounted on poles or building corners at a height of 15-20 feet. Images were captured at a frequency of 1 fps to balance data granularity and processing load under various lighting conditions, including bright daylight, dusk, and night-time with street lighting. This ensured that the edge detection techniques were tested across different environmental lighting scenarios. Images were resized to a standard resolution of 1920×1080 pixels to maintain consistency in processing. Hardware includes Intel Core i9-9900K processor is used, 32 GB DDR4 RAM, NVIDIA GeForce RTX 2080 Ti GPU. Software includes OpenCV version 4.5.1 for implementing and testing the edge detection algorithms. Other software or tools used are Python 3.8 for scripting and algorithm development. The images were self-collected using the aforementioned camera setup at multiple intersections in a metropolitan area over a period of one month. A total of 10,000 images were collected and used for the analysis.

The dataset included images with varying traffic densities, ranging from low (few vehicles) to high (heavy congestion). Images were stored in PNG format to preserve quality and support lossless compression. The standardized image size was 1920×1080 pixels, ensuring uniformity for processing. The data analysis focused on assessing the performance of each edge detection technique against various parameters, and decisions for including or excluding certain data were based on predefined criteria. Traffic images were

collected under different environmental conditions and with varying traffic densities. The performance metrics included edge sharpness, abruptness, sensitivity factors (SF), and processing time. A comparison matrix was formulated to systematically evaluate these metrics, ensuring a comprehensive analysis. Certain data points were excluded if they did not meet the quality standards required for accurate analysis, such as images with excessive noise or those taken under extreme lighting conditions that rendered the edge detection ineffective. The data was processed and analyzed using Python scripts, ensuring consistent and repeatable results. The inclusion criteria focused on images that represented typical traffic conditions, enabling a robust comparison of edge detection techniques. The findings were tabulated and analyzed to determine the most effective method for traffic signal management, with the Canny edge detection technique showing superior performance in most evaluated parameters. This step helps in minimizing false edge detection. The kernel size for Gaussian blur was empirically set to 5×5 based on preliminary experiments. For normalization all images are converted to grayscale using `cv2.cvtColor`. This step simplified the processing by reducing the complexity from three color channels to one and Resized images to a standard resolution of 1920×1080 pixels to ensure uniformity in subsequent processing. Evaluated the correctness of edge detection by evaluating observed edges with manually annotated ground truth edges for accuracy. The proportion of correctly identified edges to the total number of edges identified. The proportion of correctly identified edges to the total number of actual edges in the ground truth.

4. RESULTS AND DISCUSSION

For the sake of analyse the evaluation of edge detection technique and image segmentation technique, traffic images were collected. A comparison matrix is formulated which includes orientation, size, environmental lighting, multiple detection, traffic density, sharpness ratio (SR), abruptness, and SF are considered for the comparative analysis. Table 1 shows the comparative analysis of types of edge detection technique employed for the TMS. The first parameter for the comparison of edge detection technique is based on the size of the object detection. It can be observed that small, medium, and large size object detection is possible using all the edge detection technique. All the objects considered for the analysis are unfamiliar, ie no pre-defined features are available regarding the nature of the object. There are three subdivision of environmental lighting such as bright, enough and dull Figure 7. The Canny edge detection shows the superior performance in the three environmental lighting condition and the edges of obstacles found sharp in the case of Canny edge detection where as other edge detection techniques are not able to display the sharpness of the obstacle as shown in Figures 7(a)-7(e)

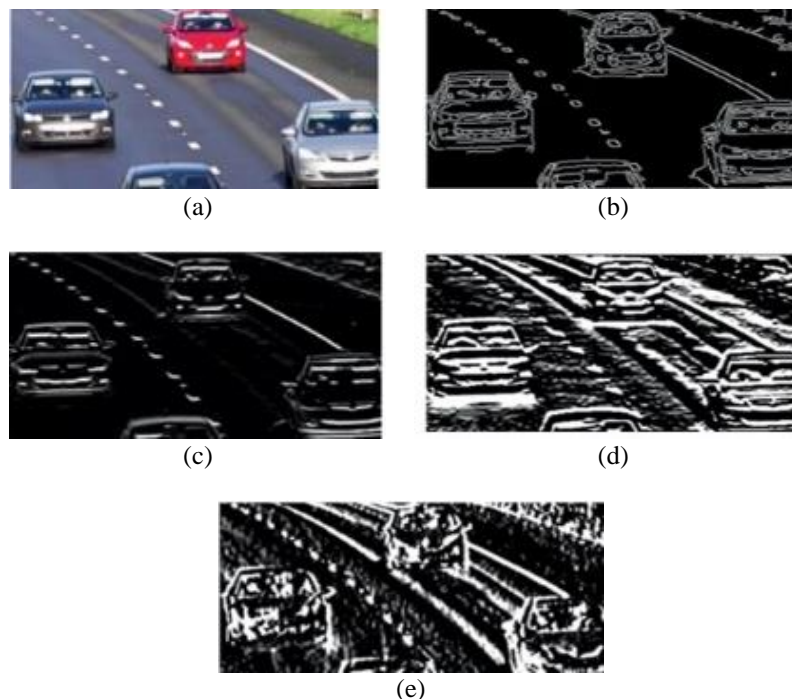


Figure 7. Edge detection comparison on environmental lighting: (a) original image, (b) Canny, (c) prewith, (d) sobel, and (e) robert edge detection techniques

Table 1. Performance comparison of edge detection in TMS

S.No.	Parameters	Canny	Prewith	Sobel	Robert
1	size	Small, medium, and large sizes	Small, medium, and large sizes	Small, medium, and large sizes	Small, medium, and large sizes
2	Unfamiliar object	yes	yes	yes	yes
3	Environmental lighting	Edges of obstacles are sharp	Edges of obstacles are not sharp	Edges of obstacles are not sharp	Edges of obstacles are not sharp
4	Multiple detection	yes	yes	yes	No
5	Average traffic density	30.12	28.45	26.61	25.88
6	Sharpness ratio	15.56	14.92	15.23	13.45
7	Abruptness	0.25	0.23	0.18	0.14
8	Sensitivity factor	0.65	0.61	0.64	0.56
9	Processing time	99 s	120 s	135 s	165 s
10	Time taken to process a frame	0.15 s	0.18 s	0.25 s	0.25 s
11	Average frame per second	15.70 FPS	12.43 FPS	10.27 FPS	9.27 FPS

Also, it is found that all the edge detection techniques are capable of tracing the multiple detection. The average traffic densities of each edge detection techniques for 10 frames are determined. The comparative analysis of edge detection in relation with the traffic density is presented in Figure 8. Traffic density refers to the volume of vehicles present on a roadway or a particular section of road during a specific period. It is quantified as the number of vehicles passing a designated point within a given time interval, frequently measured per hour.

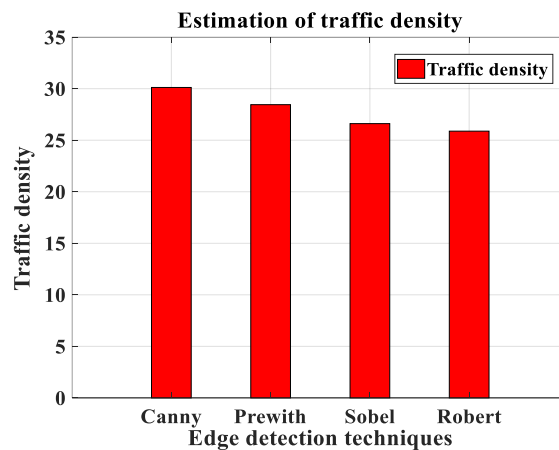


Figure 8. Traffic density in different edge detection techniques

Assessing traffic density plays a crucial role in understanding the flow of traffic and the degree of congestion on roadways. Canny edge detection can estimate the accurate traffic density compared to other edge detection techniques. The SR in image processing measures how clear and detailed an image appears. It's a way to understand how sharp or blurry the picture looks. The SR is estimated using the (10).

$$SR = \frac{\text{Highfrequencycontent}}{\text{Lowfrequencycontent}} \quad (10)$$

The higher the value of SR is desirable for image processing technique. SR gives the information about colour correction, noise reduction, and contrast enhancement. The SR of different edge detection methods are presented in Figure 9.

The value of abruptness indicates realism and image quality. The excess value of abruptness leads lose of reality. The selected edge detection techniques shows moderate value of abruptness. However, there are slight variations in the values of abruptness illustrated in Figure 10.

In image processing, SF refer to parameters that determine how responsive a particular process is to changes in input data. These factors influence how effectively an image processing technique adapts to variations in the input image. The SF is calculated using the (11).

$$SF = \frac{\text{Change in Edge Magnitude}}{\text{Change in Input Intensity}} \tag{11}$$

The comparison of edge detection technique in TMS based on SF is shown in the Figure 11.

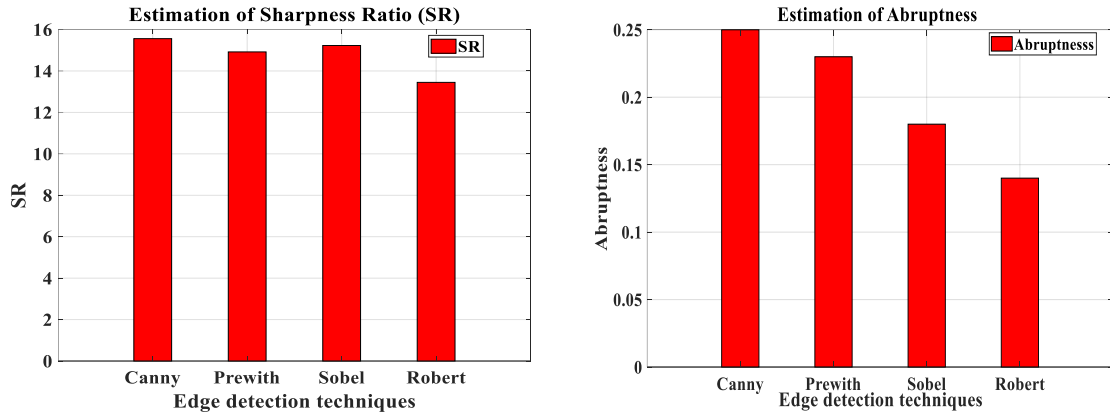


Figure 9. SR in different edge detection techniques Figure 10. SR in different edge detection techniques

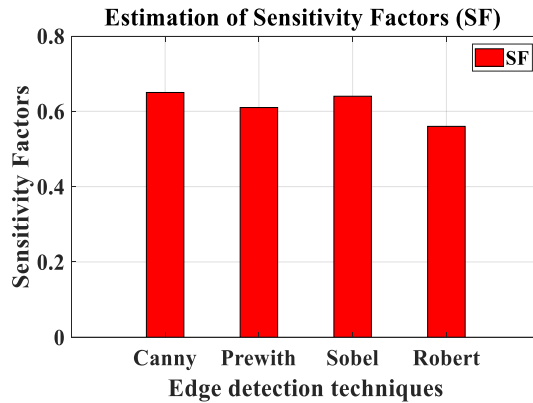


Figure 11. SF in different edge detection techniques

The Canny edge detection shows superior performance on SF. Both robert and prewith edge detection technique shows slightly less performance compared to Canny edge detection based on SF. Processing time refers to the duration required for the entire image processing task, including tasks such as noise reduction, edge detection, color correction, and other enhancements. It's crucial because shorter processing times mean faster results, that is essential for real-time applications like TMS. Reduced processing time ensures quicker feedback and enhances user experience. On the other hand, the duration required to process an individual frame, commonly known as frame processing time, is essential for tasks involving real-time video processing, such as surveillance systems, video conferencing, or augmented reality applications. For these applications, maintaining a consistent frame processing time ensures smooth video playback and responsiveness. Delays in processing time per frame can lead to laggy or choppy video output, affecting the usability and reliability of the system. The Figure 12 presents processing duration and time required to process a frame of edge detection techniques. The image processing time is less in the case of Canny edge detection technique. Also, time required to process a frame is very less in this circumstance of Canny edge detection technique. The image processing time is comparatively high in this circumstance of sobel and Robert edge detection techniques.

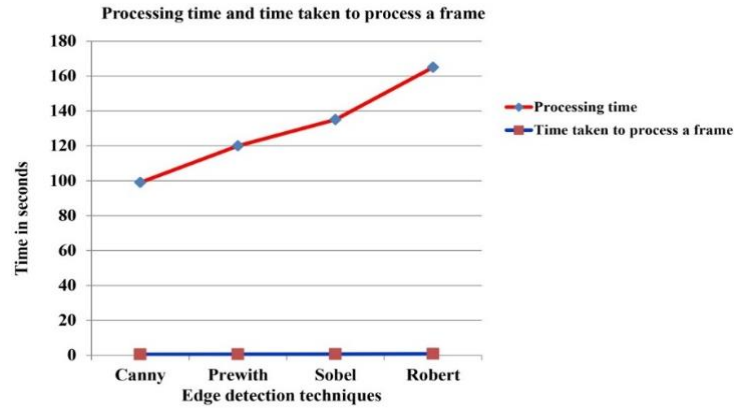


Figure 12. Processing duration and the time required to process a frame

The significance of average FPS lies in its measurement of the performance and usability of video-based systems, particularly in real-time applications. Average FPS indicates the number of frames processed and displayed per second in an image. The Figure 13 illustrates FPS of various edge detection techniques. The higher the value of FPS is desirable for the better quality of image and smoothness. In this analysis, Canny edge detection shows higher value of FPS compared to other edge detection techniques.

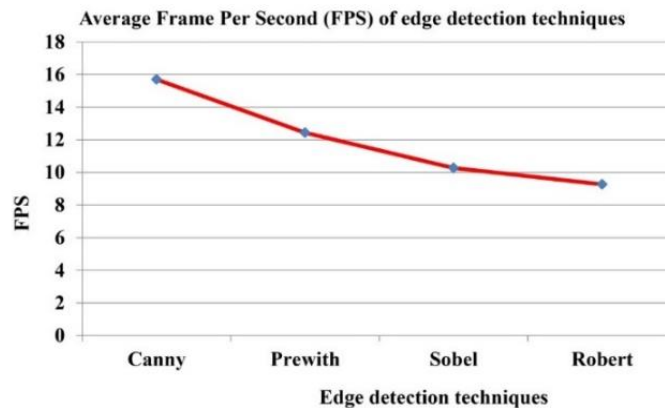


Figure 13. Processing time and time taken to process a frame

5. CONCLUSIONS




The inefficiencies in TMS result in excessive fuel consumption, infrastructure strain, decreased quality of life, negative economic impacts, and increased air pollution. These inefficiencies negatively impact both individual countries and the global community. Image processing techniques offer a promising solution to address the shortcomings of ineffective TMS. By analyzing captured images and applying sophisticated image processing algorithms, TMS authorities can plan and manage traffic systems more effectively. In this research, a detailed comparative analysis of various edge detection techniques was conducted using both qualitative and quantitative parameters. The qualitative parameters considered include environmental lighting, object detection, and object size. Quantitative parameters such as average traffic density, SR, abruptness, SF, processing time, time taken to process a frame, and average FPS were evaluated using mathematical formulas. Among the techniques analyzed, Canny edge detection demonstrated superior performance, with an average traffic density measurement of 30.12, outperforming Prewitt, Sobel, and Robert edge detection techniques. Key performance indicators such as SR, abruptness, and SF also showed better results with Canny edge detection compared to other techniques, although the differences were not significantly large. Importantly, the Canny technique exhibited the shortest processing time at 99 seconds, whereas Robert edge detection took 165 seconds, which is undesirable. The highest average FPS achieved by Canny edge detection further confirms its effectiveness in providing realistic and efficient image processing.

The findings suggest that conventional image processing techniques can be enhanced by implementing methods such as noise reduction, adaptive thresholding, precise edge localization, and improved gradient edge techniques. Future research should focus on these advanced methodologies to develop more efficient and accurate TMS, ultimately contributing to better urban mobility and reduced environmental impact. This study provides a comprehensive evaluation of edge detection techniques for TMS, emphasizing the superior performance of Canny edge detection. The results underscore the importance of continuous innovation in image processing to address the evolving challenges of traffic management and to enhance the overall effectiveness of urban transportation systems.




REFERENCES

- [1] M. Sigala, A. Beer, L. Hodgson, and A. O'Connor, *Big Data for Measuring the Impact of Tourism Economic Development Programmes: A Process and Quality Criteria Framework for Using Big Data*, 2019.
- [2] G. Nguyen *et al.*, "Machine learning and deep learning frameworks for large-scale data mining: a survey," *Artificial Intelligence Review*, vol. 52, no. 1, pp. 77–124, 2019, doi: 10.1007/s10462-018-09679-z.
- [3] C. Shorten and T. M. Khoshgoftaar, "A survey on image data augmentation for deep learning," *Journal of Big Data*, vol. 6, no. 1, 2019, doi: 10.1186/s40537-019-0197-0.
- [4] R. Vinayakumar, M. Alazab, K. P. Soman, P. Poomachandran, A. Al-Nemrat, and S. Venkatraman, "Deep learning approach for intelligent intrusion detection system," *IEEE Access*, vol. 7, pp. 41525–41550, 2019, doi: 10.1109/ACCESS.2019.2895334.
- [5] K. Sivaraman, R. M. V. Krishnan, B. Sundarraj, and S. Sri Gowthem, "Network failure detection and diagnosis by analyzing syslog and SNS data: Applying big data analysis to network operations," *International Journal of Innovative Technology and Exploring Engineering*, vol. 8, no. 9 Special Issue 3, pp. 883–887, 2019, doi: 10.35940/ijitee.I3187.0789S319.
- [6] C. B. Rafter, B. Anvari, and S. Box, "Traffic responsive intersection control algorithm using GPS data," *2017 IEEE 20th International Conference on Intelligent Transportation Systems (ITSC)*. IEEE, 2017.
- [7] W. Jiang and J. Luo, "Graph neural network for traffic forecasting: A survey," *Expert systems with applications*, vol. 207, 2022: 117921.
- [8] S. Kumar and M. Singh, "Big data analytics for healthcare industry: Impact, applications, and tools," *Big Data Mining and Analytics*, vol. 2, no. 1, pp. 48–57, 2019, doi: 10.26599/BDMA.2018.9020031.
- [9] L. M. Ang, K. P. Seng, G. K. Ijamaru, and A. M. Zungeru, "Deployment of IoV for smart cities: applications, architecture, and challenges," *IEEE Access*, vol. 7, pp. 6473–6492, 2019, doi: 10.1109/ACCESS.2018.2887076.
- [10] B. P. L. Lau *et al.*, "A survey of data fusion in smart city applications," *Information Fusion*, vol. 52, no. January, pp. 357–374, 2019, doi: 10.1016/j.inffus.2019.05.004.
- [11] Y. Wu *et al.*, "Large scale incremental learning," *Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, vol. 2019-June, pp. 374–382, 2019, doi: 10.1109/CVPR.2019.00046.
- [12] A. Mosavi, S. Shamsirband, E. Salwana, K. W. Chau, and J. H. M. Tah, "Prediction of multi-inputs bubble column reactor using a novel hybrid model of computational fluid dynamics and machine learning," *Engineering Applications of Computational Fluid Mechanics*, vol. 13, no. 1, pp. 482–492, 2019, doi: 10.1080/19942060.2019.1613448.
- [13] V. Palanisamy and R. Thirunavukarasu, "Implications of big data analytics in developing healthcare frameworks – a review," *Journal of King Saud University - Computer and Information Sciences*, vol. 31, no. 4, pp. 415–425, 2019, doi: 10.1016/j.jksuci.2017.12.007.
- [14] J. Sadowski, "When data is capital: Datafication, accumulation, and extraction," *Big Data and Society*, vol. 6, no. 1, pp. 1–12, 2019, doi: 10.1177/2053951718820549.
- [15] J. R. Saura, B. R. Herraiez, and A. Reyes-Mendez, "Comparing a traditional approach for financial brand communication analysis with a big data analytics technique," *IEEE Access*, vol. 7, pp. 37100–37108, 2019, doi: 10.1109/ACCESS.2019.2905301.
- [16] D. Nallaperuma *et al.*, "Online incremental machine learning platform for big data-driven smart traffic management," *IEEE Transactions on Intelligent Transportation Systems*, vol. 20, no. 12, pp. 4679–4690, 2019, doi: 10.1109/TITS.2019.2924883.
- [17] S. Schulz, M. Becker, M. R. Groseclose, S. Schadt, and C. Hopf, "Advanced MALDI mass spectrometry imaging in pharmaceutical research and drug development," *Current Opinion in Biotechnology*, vol. 55, pp. 51–59, 2019, doi: 10.1016/j.copbio.2018.08.003.
- [18] C. Shang and F. You, "Data analytics and machine learning for smart process manufacturing: recent advances and perspectives in the big data era," *Engineering*, vol. 5, no. 6, pp. 1010–1016, 2019, doi: 10.1016/j.eng.2019.01.019.
- [19] Y. Yu, M. Li, L. Liu, Y. Li, and J. Wang, "Clinical big data and deep learning: Applications, challenges, and future outlooks," *Big Data Mining and Analytics*, vol. 2, no. 4, pp. 288–305, 2019, doi: 10.26599/BDMA.2019.9020007.
- [20] M. Huang, W. Liu, T. Wang, H. Song, X. Li, and A. Liu, "A queuing delay utilization scheme for on-path service aggregation in services-oriented computing networks," *IEEE Access*, vol. 7, pp. 23816–23833, 2019, doi: 10.1109/ACCESS.2019.2899402.
- [21] G. Xu, Y. Shi, X. Sun, and W. Shen, "Internet of things in marine environment monitoring: A review," *Sensors (Switzerland)*, vol. 19, no. 7, pp. 1–21, 2019, doi: 10.3390/s19071711.
- [22] M. Aqib, R. Mehmood, A. Alzahrani, I. Katib, A. Albeshri, and S. M. Altowaijri, *Smarter traffic prediction using big data, in-memory computing, deep learning and gpus*, vol. 19, no. 9, 2019.
- [23] S. Leonelli and N. Tempini, *Data Journeys in the Sciences*, 2020.
- [24] N. Stylos and J. Zwiegelaar, *Big Data as a Game Changer: How Does It Shape Business Intelligence Within a Tourism and Hospitality Industry Context?* 2019.
- [25] Q. Song, H. Ge, J. Caverlee, and X. Hu, "Tensor completion algorithms in big data analytics," *arXiv*, vol. 13, no. 1, 2017.




BIOGRAPHIES OF AUTHORS

Niranjana Chandrasekara Bharathy    is an assistant professor in the Electronics and Communication Department at Sri Venkateshwara College of Engineering, Bangalore. She completed her B.E. in Electrical and Electronics Engineering at Avinashilingam University, Coimbatore. She pursued her M.E. in embedded system technologies from Kumaraguru College of Technology, Coimbatore. With 4 years of teaching and 1 year of industry experience, she has published 5 journal papers and 5 Conference Papers, 2 Book Chapters, and 1 patent. She can be contacted at email: niranjana3194@gmail.com.






Manju Shivanna Dasappanavar    is an assistant professor in the Department of Electronics and Communication Engineering at Sri Venkateshwara College of Engineering in Bengaluru. He graduated from V.T.U. Belgaum in 2009 with a B.E. in Electronics and Communication Engineering and in 2014 with an M.Tech. in Master of Technology in Digital Communication and Networking Engineering. He has worked as a teacher for almost six years and in the industry for four years. In addition to two patents, he has published five journal papers, five conference papers, one textbook, and one book chapter in Springer through a variety of indexed journals and conferences. AI, machine learning, data science, data analytics, networks, internet of things, deep learning, and embedded systems are among his areas of interest. He can be contacted at email: manju.d_ace@svcengg.edu.in.



Anne Gowda Aleri Byre Gowda    completed his B.Tech. in Telecommunication Engineering in 2011 and M.Tech in signal processing in 2014. He has more than 9 years of experience in teaching. Now he is pursuing his Ph.D. in array signal processing and wireless communication. His area of interest includes digital signal processing and digital image processing and wireless communication. He can be contacted at email: annegowda.ace@gmail.com.



Dr. Jijesh Jisha Janardhanan    is a professor and head of the Electronics and Communication Department at Sri Venkateshwara College of Engineering, Bangalore. He earned his B.Tech. in Electronics and Communication Engineering from Kannur University, M.Tech. in Electronics from VTU, and a Doctorate in Wireless Communication from VTU. With 17 years of teaching and 5 years of research experience, he has published over 57 articles in various indexed journals and conferences. His research interests include Wireless Communication, sensor networks, Embedded Systems, and IoT. He holds one patent and is an active IEEE member, has coordinated international conferences, and supervises Ph.D. scholars at VTU. He can be contacted at email: jijesh_jj@yahoo.co.in.