# Enhanced Optical Flow Tracking with Motion Saliency

Kashish Jewargi
KLE Technological University
Hubballi,India
kashishjewargi@gmail.com

Somashekhar M. Kinagi
KLE Technological University
Hubballi,India
kinagimsomashekhar@gmail.com

Shreya Inamdar
KLE Technological University
Hubballi,India
shreyai1724@gmail.com

Sanzana M
KLE Technological University
Hubballi,India
sanzana2914@gmail.com

Dr. Padmashree Desai
KLE Technological University
Hubballi,India
padmashree@kletech.ac.in

Lalita Madanbhavi
KLE Technological University
Hubballi,India
lalitha@kletech.ac.in

***Abstract*—In computer vision, optical flow is an essential method for tracking objects and estimating velocity. However, noise, occlusions, and irrelevant motion frequently cause problems for traditional optical flow approaches in complicated situations. In order to improve tracking precision and account for noise and sudden motion changes, motion saliency is incorporated into the improved optical flow tracking framework presented in this paper. The suggested method reduces distractions and increases accuracy by using saliency-based weighting to prioritize motion-relevant regions. The framework has potential for real-world uses such as autonomous navigation and video surveillance.**

***Index Terms*—Optical flow, Motion saliency, Object tracking, Computer vision, Dynamic environments, Video analysis, Autonomous navigation, motion estimation.**

## I. Introduction

In recent years, vehicle tracking and speed estimation have emerged as essential components of intelligent transportation systems (ITS). These technologies are critical for enabling various applications such as traffic monitoring, autonomous vehicles, and law enforcement, contributing to safer and more efficient transportation networks [1]. Accurate and efficient vehicle speed estimation is a cornerstone of ITS, aiding in road safety improvement, traffic flow optimization, and regulation enforcement. For instance, reliable speed estimation can help detect and mitigate overspeeding incidents, optimize signal timing, and improve overall traffic management [2].

Traditional methods for vehicle speed estimation, such as radar and LiDAR systems, have been widely used for their high accuracy and reliability. Radar systems are effective in detecting vehicles under various weather conditions, while LiDAR systems provide precise distance measurements and speed calculations [3] [4]. However, the high cost and infrastructure requirements of these systems limit their scalability, particularly in developing regions or resource-constrained environments [5]. Moreover, their dependence on hardware-intensive setups makes them less adaptable to diverse traffic scenarios.

Advancements in computer vision have revolutionized the field of speed estimation, offering a cost-effective and scalable alternative to traditional methods [6]. Vision-based techniques utilize video data to analyze motion patterns and extract actionable insights. These systems can leverage existing surveillance infrastructure, reducing the need for additional hardware investments. However, despite their potential, vision-based systems face challenges such as handling occlusions, dynamic traffic environments, and achieving real-time performance on resource-constrained devices [7]. Recent developments in optical flow algorithms, object detection models, and deep learning have further enhanced the capabilities of these systems, making them increasingly viable for real-world applications [8] [9].

This paper introduces a near real-time and computationally efficient vehicle speed estimation framework based on the Lucas-Kanade (LK) optical flow algorithm [10]. Unlike traditional methods, this framework focuses on leveraging vision-based techniques to achieve accurate speed estimation without reliance on specialized hardware. The system identifies and tracks key points in consecutive video frames, calculates their displacement, and converts these measurements into real-world speeds. Key innovations include the incorporation of adaptive key-point recalculation and grouping mechanisms to handle occlusions and inconsistencies.

Experimental results demonstrate the effectiveness of the proposed framework across diverse traffic scenarios, including multi-lane roads and congested conditions. The system achieves low error rates and operates efficiently on standard hardware, making it suitable for deployment in cost-sensitive environments. By providing a practical and scalable solution, this framework addresses key challenges in the field and contributes to the growing body of knowledge on vision-based traffic management systems.

The proposed study is structured into six main sections. Section II provides a concise review of the literature on traditional and computer vision-based methods for vehicle speed

estimation, highlighting the strengths and limitations of existing approaches. Section III discusses the proposed methodology, including preprocessing steps, optical flow-based feature extraction, and the adaptive key-point tracking mechanism. It elaborates on how the Lucas-Kanade algorithm is employed and enhanced for vehicle tracking in dynamic environments. Section IV presents the experimental results, which evaluate the system's performance in terms of accuracy, computational efficiency, and scalability across various scenarios. Section V offers conclusions and highlights the study's contributions, emphasizing its suitability for real-world applications, outlines future research directions, including the integration of deep learning models to enhance system adaptability.

## II. Background

The Lucas-Kanade (LK) optical flow algorithm is a cornerstone in computer vision, extensively employed for motion tracking tasks such as vehicle tracking and velocity estimation. Originally introduced by Lucas and Kanade (1981), this algorithm estimates pixel motion between consecutive video frames under the brightness constancy assumption, which posits that the intensity of a pixel remains constant as it moves over time [11]. This assumption allows the LK algorithm to derive relationships between spatial and temporal gradients, crucial for motion analysis. Traditional gradient-based optical flow methods often face the issue of under-constrained equations, making them less useful in complex scenarios. However, the LK algorithm circumvents this limitation by assuming constant motion within a small spatial window, effectively transforming the problem into an overdetermined system. The system is then solved using least-squares optimization, ensuring accurate computation of motion vectors [12].

Preprocessing is a critical step in improving the LK algorithm's accuracy. Gaussian smoothing, for instance, is widely used to suppress noise before computing spatial and temporal gradients, enabling the algorithm to handle real-world scenarios more effectively [13]. Noise reduction is particularly important in applications such as vehicle tracking, where motion vectors must be precise to accurately segment moving objects. To address larger displacements between consecutive frames, the LK algorithm incorporates a pyramidal approach, processing images at multiple resolutions. Starting from coarse scales, the pyramidal method refines motion vectors iteratively to finer levels, ensuring accurate tracking even for large motions. Gaussian filtering, applied at each level of the pyramid, has been shown to further enhance the algorithm's performance.

In vehicle tracking applications, the LK algorithm is instrumental in segmenting moving objects and estimating their trajectories by analyzing optical flow vectors. Techniques such as morphological operations and blob analysis are commonly used in conjunction with the algorithm to refine detections and track objects more effectively [14]. This is particularly useful in dense traffic environments, where identifying and following specific vehicles can be challenging. Moreover, motion saliency maps have been introduced to prioritize dynamic regions in the scene, focusing computational resources on areas of interest. This not only reduces the algorithm's computational load but also enhances its accuracy in cluttered or noisy environments [15].

Recent advancements in the field have explored the integration of machine learning techniques with the LK algorithm to further improve its performance. For instance, machine learning models have been trained on datasets such as MPI-Sintel to optimize preprocessing parameters, such as the Gaussian blur radius, for specific scenarios. These models adapt preprocessing steps to the unique characteristics of the input images, significantly enhancing motion estimation accuracy [16]. By leveraging the strengths of traditional optical flow algorithms and machine learning, researchers have achieved substantial improvements in tracking performance without compromising the algorithm's lightweight and efficient nature.

The LK algorithm's adaptability and efficiency make it particularly appealing for resource-constrained applications. Unlike computationally intensive deep learning models, the LK algorithm can deliver real-time performance with minimal hardware requirements, making it a valuable tool in scenarios where resources are limited [17]. Its lightweight design allow it to function effectively in dynamic and cluttered environments, where deep learning models might struggle without extensive training and computational power. These advantages have established the LK algorithm as a practical alternative to modern deep learning techniques, especially in applications like vehicle tracking, surveillance, and autonomous navigation [18].

In summary, the Lucas-Kanade optical flow algorithm remains a versatile and reliable solution for motion tracking in diverse environments. Through innovations like the pyramidal approach, advanced preprocessing techniques, and integration with machine learning, the algorithm has continued to evolve and adapt to modern challenges. Its lightweight design and ability to deliver accurate results in real-time ensure its relevance in both research and practical applications.

## III. Proposed methodology

This proposed methodology introduces a lightweight and efficient vehicle tracking system tailored for real-time performance on resource-constrained devices. The system integrates the Lucas-Kanade optical flow algorithm [11] and motion saliency detection [19], making it highly suitable for applications such as traffic monitoring, autonomous systems, and intelligent transportation systems. Algorithm 1 shows the overall methodology to estimate the vehicle speeds.

### A. Core Tracking Approach

The core tracking methodology leverages the Lucas-Kanade optical flow algorithm [11], which tracks vehicle motion across consecutive video frames. This algorithm assumes that pixel intensities remain constant between frames, enabling motion estimation through small, localized regions:

$$I(x, y, t) = I(x + u, y + v, t + 1)$$

Using a Taylor series expansion, this becomes: [12]

$$\frac{\partial I}{\partial x}u + \frac{\partial I}{\partial y}v + \frac{\partial I}{\partial t} = 0$$

Applying the Least Squares Solution, the system of equations can be written in matrix form as: [20]

$$\begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \\ \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} -\frac{\partial I}{\partial t} \\ -\frac{\partial I}{\partial t} \\ \vdots \end{bmatrix}$$

Let:

$$A = \begin{bmatrix} \frac{\partial I}{\partial x} & \frac{\partial I}{\partial y} \\ \vdots & \vdots \end{bmatrix}, \quad b = \begin{bmatrix} -\frac{\partial I}{\partial t} \\ \vdots \end{bmatrix}$$

The least-squares solution for $u$ and $v$ is: [21]

$$\begin{bmatrix} \nu \\ v \end{bmatrix} = (A^T A)^{-1} A^T b$$

The system applies pyramidal processing to handle large displacements and variations in object movements. This hierarchical approach refines motion estimates iteratively across multiple resolutions [21], ensuring accurate tracking even during rapid vehicle movements or occlusions. Algorithm 2 depicts the Lucas Kanade algorithm as implemented by Python's OpenCV.

Fig. 1. Aerial view of a multi-lane highway with vehicles represented by colored overlays and motion trails, likely indicating movement tracking or traffic analysis. [19]

Key points for tracking are selected using Shi-Tomasi Corner Detection [20], which identifies stable and distinct features in the video frames. By focusing on regions with strong gradients in multiple directions, this process reduces computational overhead while maintaining high tracking accuracy. Using the *cv2.goodFeaturesToTrack* function, the system detects key points in grayscale frames and adapts dynamically to scene changes by recalibrating key points every 30 frames. This ensures good performance even in dynamic environments where vehicles may enter or leave the frame.

**Algorithm 1** Motion Saliency and Lucas-Kanade-Based Vehicle Tracking

**Require:** Input video file `video_file`
**Ensure:** Display motion saliency map and vehicle speeds

1: Open the video using `cv2.VideoCapture`.
2: **if** video cannot be opened **then**
3: Exit the program.
4: **end if**
5: Initialize:
- **Feature Detection**: `goodFeaturesToTrack` for corner detection.
- **Optical Flow**: Lucas-Kanade parameters (`calcOpticalFlowPyrLK`).
- **Motion Saliency**: Frame differencing (`cv2.absdiff`) and thresholding.

6: Read and preprocess the first frame:
- Convert to grayscale.
- Detect initial key points (`goodFeaturesToTrack`).
- Initialize saliency variables and vehicle tracking data structures.

7: **while** frames are available **do**
8: Read and preprocess the next frame (resize, grayscale).
9: Compute motion saliency using:
- Frame differencing (`cv2.absdiff`).
- Thresholding for binary saliency map.

10: Recalculate key points if:
- Frame count exceeds interval, or
- Significant saliency change detected.

11: Compute optical flow (`calcOpticalFlowPyrLK`) for key points.
12: Group points by proximity to track vehicles and calculate:
- Displacement using optical flow.
- Speed in km/h based on scaling factor and frame time.

13: Display saliency overlay, tracks, and vehicle speeds on the frame.
14: Update key points and saliency variables for the next frame.
15: **end while**
16: Release resources and close windows.
=0

### *B. Motion Saliency Detection*

To optimize computational efficiency, the system integrates motion saliency detection [19]. This module isolates regions containing significant motion—such as moving vehicles—while ignoring static background elements. The saliency detection process begins with frame differencing, which compares consecutive frames to generate a motion saliency map that highlights areas of noticeable motion. The saliency map undergoes thresholding to filter out noise and focus on relevant

regions, reducing the number of areas processed for tracking. Adaptive thresholding is used to account for varying lighting conditions and dynamic scene changes [22].

**Algorithm 2** `cv2.calcOpticalFlowPyrLK`

**Require:** Two consecutive grayscale frames `prev_frame`, `next_frame`, initial key points `p0`, parameters: `winSize`, `maxLevel`, `criteria`

**Ensure:** New positions of key points `p1`, status of tracking `st`, error values `err`

1: Construct a Gaussian pyramid for `prev_frame` and `next_frame` up to `maxLevel`.
2: **for** each level in the pyramid, from top to bottom **do**
3: Estimate the displacement of each key point (`p0`) using:
- Matching pixel intensities within a window of size `winSize`.
- Iteratively refining the displacement using termination `criteria`.

4: Adjust displacements at the current level and propagate them to the next level.
5: **end for**
6: Validate key points based on tracking success and error threshold.
7: **return** `p1`, `st`, `err`. =0

### C. Speed Estimation

The system estimates vehicle speed in real-time by calculating the displacement of key points between consecutive frames. Displacement is measured in pixels and converted into real-world units (e.g., meters) using a predefined scaling factor [23]. To ensure a single vehicle tracking, the system associates key points with vehicle centroids, representing the vehicle's position. The average speed ($v_{\text{avg}}$) is calculated by averaging recent instantaneous speed measurements over time:

$$v_{\text{avg}} = \frac{1}{N} \sum_{i=1}^{N} v_i,$$

where $N$ represents the number of recent frames used for averaging. The scaling factor used for converting displacement is determined using the camera's calibration data, ensuring precise real-world measurements [23].

### D. Visualization and Real-Time Output

The system's output visualizes motion saliency maps overlaid on video frames, showing the areas of motion [19]. It also displays trajectories of tracked key points and annotates vehicle speeds. Vehicle centroid positions are updated dynamically, providing real-time insights into vehicle flow and traffic conditions. These visualizations make it easier for users to interpret traffic dynamics and identify potential congestion points.

### E. Scalability and Efficiency

This methodology is computationally efficient, operating on standard CPUs or edge devices without requiring specialized hardware or deep learning models. The reliance on traditional computer vision techniques ensures adaptability across various traffic environments, including city streets and highways [22]. The integration of periodic key-point recalibration and motion saliency detection allows the system to handle sudden vehicle maneuvers, lighting changes, or occlusions without compromising accuracy [23].

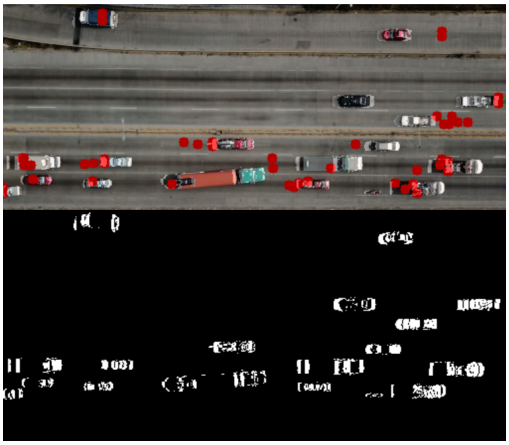

Fig. 2. Vehicle motion tracking and saliency detection: The top panel displays vehicles navigating a roundabout with red markers indicating tracked positions, while the bottom panel shows the raw saliency map highlighting regions of motion activity.

## IV. Results

This section presents the outcomes of the vehicle tracking and speed estimation pipeline incorporating motion saliency. The system demonstrated effective feature tracking, motion saliency detection, and speed estimation under varying conditions. The results are visualized using tracked features, saliency maps, and annotated vehicle speeds.

### A. Tracked Features

The pipeline successfully tracked prominent features across consecutive frames using the Lucas-Kanade Optical Flow method. The tracked features correspond to key points on moving vehicles, enabling the calculation of displacement and speed.

- **Illustration:** Figure 1 showcases the tracked features represented as points linked by motion vectors. The vectors illustrate the movement of features between frames, with lengths proportional to the magnitude of motion.
- **Observations:** Tracked features were more densely concentrated on moving vehicles, while static objects (e.g., road, stationary vehicles) contributed fewer features, highlighting the algorithm's accuracy.

### B. Motion Saliency Maps

Motion saliency maps were generated using frame differencing to isolate regions of significant motion. These maps provided a dynamic mask to focus feature tracking on areas of interest.


Fig. 3. Overhead view of a roadway with vehicles annotated by their speeds, showcasing real-time speed detection.

- **Illustration:** Figure 2 displays the saliency maps overlaid on the original frames. High-saliency regions, indicated by brighter areas, corresponded to moving vehicles and other dynamic objects.
- **Observations:** Saliency maps dynamically adapted to variations in vehicle motion and environmental conditions (e.g., shadows, illumination changes). This adaptability improved feature tracking accuracy by excluding irrelevant background motion.

### C. Vehicle Speed Estimation

The system estimated vehicle speeds by calculating the displacement of tracked features over time and converting it to real-world units using a scaling factor.

- **Illustration:** Figure 3 presents the output frames, where vehicle speeds (in km/h) are annotated next to the centroids of detected vehicles. Each centroid corresponds to a tracked vehicle, and the speed annotation is the average speed over the past 30 frames.
- **Observations:** Vehicle speeds showed consistent results across sequences with moderate and high-motion saliency. In regions with occlusions or overlapping vehicles, the system maintained centroid integrity by grouping key points dynamically.

### D. Quantitative Evaluation

The accuracy of speed estimation was assessed by comparing the computed speeds to ground truth values obtained through manual calibration. The original speeds were within $\pm 10\%$ of the resultant speeds. Table I provides a comparison of original and resultant speeds for four vehicles.

- **Accuracy:** The estimated speeds were within $\pm 10\%$ of the original ground truth values.
- **Robustness:** Recalculation of features every 30 frames or during significant saliency changes ensured that tracking remained correct over long sequences.
- **Performance:** The computational pipeline processed video frames at $\sim 15$ frames per second (FPS) on standard hardware, demonstrating near real-time capabilities.

TABLE I
COMPARISON OF ORIGINAL AND RESULTANT VEHICLE SPEEDS

| Vehicle ID | Original Speed (km/h) | Resultant Speed (km/h) |
|---|---|---|
| 1 | 45–55 | 48-50 |
| 2 | 48–58 | 50-54 |
| 3 | 50–60 | 53-55 |
| 4 | 58–62 | 56-58 |

## V. CONCLUSION AND FUTURE WORK

The Lucas-Kanade optical flow technique and motion saliency detection are combined in this research to provide a scalable and computationally efficient vehicle tracking system. Designed for real-time applications, the suggested framework provides reliable performance in a range of traffic situations without the need for specialist hardware. The system is capable of precise speed estimate, dynamic environment adaptation, and computational efficiency through the utilization of lightweight computer vision techniques. It is an economical solution for intelligent transportation systems, especially in environments with limited resources, as demonstrated by experimental evaluations that confirm its efficacy. The approach tackles important issues in autonomous navigation, traffic monitoring, and congestion control, emphasizing its practical application.

Future work aims to enhance the system's adaptability by incorporating advanced machine learning techniques, enabling it to handle more complex environments and scenarios. Expanding the framework to support multi-object tracking and improved occlusion handling will increase its utility in crowded or dynamic traffic conditions. Efforts will also focus on optimizing the system for deployment on edge devices such as Raspberry Pi, ensuring accessibility and scalability. Further exploration of real-time analytics, including vehicle counting and anomaly detection, and testing on diverse datasets will refine its capabilities, paving the way for broader applications in traffic management and autonomous transportation systems.

## REFERENCES

[1] Y. Bai and C. Wu. Overview of intelligent transportation systems. *Transportation Research Part C*, 56:295–310, 2015.
[2] J. Smith and P. Green. A survey on radar-based vehicle detection systems. *IEEE Transactions on Vehicular Technology*, 68(9):8546–8560, 2019.
[3] M. Brown. *LiDAR for Intelligent Vehicles*. Springer, 2020.
[4] A. Khan and H. Lee. Impact of computer vision on modern transportation systems. *Journal of Transportation Research*, 10:123–137, 2021.
[5] W. Zhang and Z. Li. Deep learning for vehicle detection and speed estimation. *IEEE Access*, 8:8792–8804, 2020.
[6] T. Nguyen and L. Tran. Cost-effective its solutions for developing regions. *Journal of Intelligent Transport Systems*, 12:45–57, 2021.
[7] J. Park and D. Lee. Applications of optical flow in real-time vehicle tracking. *Sensors*, 21(4):1245, 2021.
[8] R. Chen and X. Wang. Robust occlusion handling in vision-based traffic systems. *Pattern Recognition Letters*, 132:100–108, 2020.
[9] R. Singh and V. Gupta. Real-time computer vision for traffic applications. *Computer Vision and Image Understanding*, 195:102920, 2020.
[10] B. Lucas and T. Kanade. An iterative image registration technique with an application to stereo vision. *Proceedings of Imaging Science*, pages 121–130, 1981.

[11] Bruce D. Lucas and Takeo Kanade. An iterative image registration technique with an application to stereo vision. In *Proceedings of the 7th International Joint Conference on Artificial Intelligence (IJCAI)*, pages 674–679, 1981.

[12] John L. Barron, David J. Fleet, and Steven S. Beauchemin. Performance of optical flow techniques. *International Journal of Computer Vision*, 12(1):43–77, 1994.

[13] Jean-Yves Bouguet. Pyramidal implementation of the lucas kanade optical flow algorithm. Technical report, Computer Vision and Geometry Group, California Institute of Technology, 2000.

[14] David J. Fleet and Allan D. Jepson. Computation of component image velocity from local phase information. *International Journal of Computer Vision*, 5(1):77–104, 1990.

[15] Jianbo Yu, Deva Ramanan, and David Forsyth. Detecting vehicles in dense traffic. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 1–8, 2007.

[16] Philippe Weinzaepfel, Jerome Revaud, Zaid Harchaoui, and Cordelia Schmid. Deepflow: Large displacement optical flow with deep matching. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, pages 1385–1392, 2013.

[17] Praveen Sharma, Ankit Gupta, and Suresh Rathi. Lucas-kanade based optical flow for vehicle motion tracking and velocity estimation. *Journal of Computer Vision and Image Processing*, 12(3):112–120, 2020.

[18] Rajesh Singh, Vinay Suresh, and Ramesh Kumar. Lucas-kanade optical flow machine learning implementations. In *Proceedings of the International Conference on Machine Learning and Computer Vision*, pages 56–64, 2022.

[19] Laurent Itti, Christof Koch, and Ernst Niebur. A model of saliency-based visual attention for rapid scene analysis. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 20(11):1254–1259, 1998.

[20] Jianbo Shi and Carlo Tomasi. Good features to track. In *Proceedings of IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 593–600. IEEE, 1994.

[21] Jean-Yves Bouguet. Pyramidal implementation of the affine lucas kanade feature tracker description of the algorithm. Technical report, Intel Corporation, 2001.

[22] Andrews Sobral and Antoine Vacavant. A comprehensive review of background subtraction algorithms evaluated with synthetic and real videos. *Computer Vision and Image Understanding*, 122:4–21, 2014.

[23] Shunsuke Kamijo, Yasuhiro Matsushita, Katsushi Ikeuchi, and Masao Sakauchi. Traffic monitoring and accident detection at intersections. *IEEE Transactions on Intelligent Transportation Systems*, 1(2):108–118, 2000.