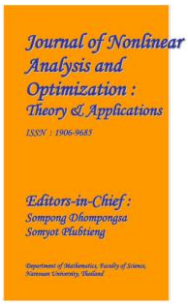
Journal of **N**onlinear **A**nalysis and **O**ptimization
Vol. 15, Issue. 1, No.10 : 2024
ISSN :**1906-9685**



METHODOLOGY FOR VEHICLE DETECTION AND COUNTING USING PYTHON

[#1]S.Sai Bhargavi, [2]T.sucharita, [3]V.V.R Murthy, [4]V.chandu, and [5]S.lavanya
[1,2,3,4 &5] UG Students, Department of Civil Engineering, Vignan's Institute of Information Technology (A), Visakhapatnam, India.

[#6]Mr.B.Barhmaiah, Assistant Professor, Department of Civil Engineering, Vignan's Institute of Information Technology (A), Visakhapatnam, India.

ABSTRACT: Vehicle recognition and counting are essential for intelligent transportation systems to manage traffic efficiently. The researcher's manual methods have been used for years to extract the traffic data from the recorded videos**.** Further Image Processing, Sensor Networks, GIS and Mapping Technologies, and Bluetooth and Wi-Fi Tracking techniques are used to collect or extract traffic data. Computer vision techniques and virtual detection zones provide accurate vehicle identification. Real-time video-based traffic flow monitoring, planning, and control solutions are made possible by Open CV, YOLO, virtual detectors, and blob tracking technologies. This study aimed to develop a methodology to extract classified traffic volume using Python programming. Open CV is used to detect the objects and count the total objects. In this study, we have developed a remedy for the high traffic issue using video surveillance and considering the video data from the traffic cameras. We used a Create background subtract method, an adaptive thresholding strategy, and a virtual detector. Python's Open CV tool was used to implement the system. The proposed method shows the 78% accuracy in data extraction when evaluated with manual data extraction. Additionally, the paper discusses potential applications, challenges, and future research directions in vehicle detection using Python-based technique
Keywords: Open CV, Python Programming, traffic data extraction, Vehicle Detection.

---

1. INTRODUCTION

In the realm of transportation engineering, the efficient management of traffic flow is a critical endeavor, essential for ensuring road safety, minimizing congestion, and optimizing infrastructure utilization. One of the key aspects of this management is the accurate detection and counting of vehicles, which forms the foundation of various traffic management systems. Traditional methods of vehicle detection and counting often involve manual observation or simplistic sensor-based systems. However, these methods are limited in their scope, accuracy, and scalability. With the advent of computer vision and machine learning technologies, there has been a paradigm shift in the way traffic data is collected and analyzed.

Traffic data extraction systems use various techniques to gather, process, and analyze traffic-related information. These techniques can be broadly categorized into the following:

1. **Video Analytics:** Utilizes computer vision algorithms to extract traffic data from video feeds. Techniques such as object detection, tracking, and classification are employed to identify vehicles, pedestrians, and other objects on the road.
2. **Sensor Networks:** Includes technologies like Inductive Loop Detectors, Piezoelectric Sensors, and Magnetic Sensors placed on roads to detect the presence and movement of vehicles. These sensors can provide real-time traffic flow information.
3. **GPS and Mobile Data:** Uses GPS devices in vehicles and mobile phones to track the movement and speed of vehicles. This data can be anonymized and aggregated to analyze traffic patterns.
4. **Crowd sourcing:** Involves collecting traffic data from users through mobile apps or social media platforms. Users can report traffic conditions, accidents, or road closures, providing real-time information to traffic management systems.
5. **Radar and LiDAR Technology:** Radar and LiDAR sensors can provide detailed information about vehicle speed, distance, and direction, enabling precise traffic monitoring and management.
6. **Machine Learning and AI:** These technologies are used to analyze and predict traffic patterns based on historical data. They can be used to optimize traffic signal timings, predict congestion, and suggest alternative routes.
7. **Data Fusion:** Combines data from multiple sources, such as video feeds, sensor networks, and GPS data, to provide a comprehensive view of traffic conditions. Data fusion techniques help improve the accuracy and reliability of traffic data extraction system

This research paper presents a methodology for vehicle detection and counting using Python programming language, with a focus on leveraging the capabilities of OpenCV (Open Source Computer Vision Library). OpenCV provides a robust set of tools and algorithms for image processing and computer vision, making it ideal for developing sophisticated vehicle detection systems. The methodology outlined in this paper aims to address the limitations of traditional methods by providing a more accurate and efficient way to detect and count vehicles in various traffic scenarios. By utilizing OpenCV's capabilities, the proposed methodology offers a scalable and reliable solution that can be easily integrated into existing traffic management systems.

2. REVIEW OF LITERATURE

Several studies have investigated vehicle detection and counting systems, employing various technologies and approaches. Chen et al. (2017) described a system combining deep learning and computer vision to recognize and count in complex traffic settings accurately. Zhang et al. (2019) investigated radar-based sensors for detecting and tracking cars, maintaining consistent performance in all weather conditions. Li et al. (2018) developed a system that uses camera and LiDAR technology to produce exact findings, particularly in metropolitan areas. Chen et al. (2016) studied machine learning approaches for adaptive vehicle recognition and counting. Furthermore, Gupta and Tyagi (2014) used GPS and GIS technologies to quickly identify and tally, highlighting the breadth of options available for improving traffic management systems. Barhmaiah Borigarla and S. Moses Santhakumar conducted research on signalized crossings and traffic circumstances, which advanced transportation engineering. In 2022, they presented delay models for signalized intersection lane allocations in a variety of traffic scenarios, exposing traffic flow dynamics. Their research on signalized crossings with vehicle-actuated controlled systems in many traffic scenarios demonstrates their traffic engineering skills (Borigarla et al., 2022b). Their work with M-Sand and quarry dust on stiff pavements demonstrates their interdisciplinary approach to infrastructure development (Borigarla et al., 2022c). Barhmaiah Borigarla, Triveni Buddaha, Sai Kiran, and G. Pritam Hait additionally conducted an experimental study on pavement materials to demonstrate their dedication to sustainable and creative construction procedures (Borigarla et al., 2022c). They are assisted by M. L. L. Priyanka, M. Padmakar, and B. Barhmaiah, who researched rural road development and the value of rural infrastructure

(Priyanka et al., 2020). B. Brahmaiah, K. Srinivas, and A. Devi Prasad's study on modeling route choice behavior in urban transit systems provides a behavioral perspective on transportation planning and throws light on commuter decision-making (Brahmaiah et al., 2017).

Yaqoob and Shahid (2020) used deep learning, precisely the YOLO technique, to recognize objects in real-time with high accuracy and efficiency. Singh et al. (2018) investigated image-processing techniques for identifying and quantifying cars, focusing on urban traffic conditions. Kim et al. (2016) described a vehicle recognition system that uses sensor fusion to merge data from cameras, LiDAR, and radar sensors. This technique allows for exact tracking in various locations. Bhuyan and Bordoloi (2015) developed a Bluetooth and Wi-Fi tracking system to identify and count vehicles, demonstrating its effectiveness in traffic surveillance. Ding et al. (2019) studied the use of deep learning approaches, specifically a mix of Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs), to improve vehicle detection accuracy in surveillance footage. These studies focus on the many strategies and technologies that can be utilized to enhance traffic management systems. simulating and analyzing various traffic dynamics and recommending alternative routes. Marisamynathana P. Vedagiri (2016) suggested a unique pedestrian delay model for signalized crossroads, which was validated using field data obtained at the Holkar junction in Mumbai, India. Cao, Yun, Zhong, and Huang (2016) described a method for recognizing automobiles that combines road line identification and background subtraction. The Local Features Method (2015) provided an alternate technique in which objects' features were classified into discrete categories using categorization models. Sokalski et al. (2010) used color features to discern between manufactured and natural goods. Asifa Mehmood Qureshi and Ahmed Jalal (2023) developed a model for identifying and monitoring autos in aerial pictures of roundabouts, with outstanding recognition and tracking accuracy. In their 2010 paper, Sahil et al. presented a system that uses components to recognize cars in low-resolution aerial data. The study focuses on applying Scale-Invariant Feature Transform (SIFT) highlights. These studies help to enhance techniques for recognizing automobiles and regulating traffic. These studies demonstrate the vast diversity of methods and technology used for vehicle detection and counting. While each strategy has advantages and disadvantages, the ultimate goal is to create more efficient and dependable traffic management systems.

3. METHODOLOGY

Detecting the direction of vehicles using Python programming typically involves computer vision techniques, particularly object detection and tracking. Here is a high-level methodology to detect the direction, type of vehicles, and volume using Python. Figure 1 shows the methodology for computer vision techniques for traffic data extraction

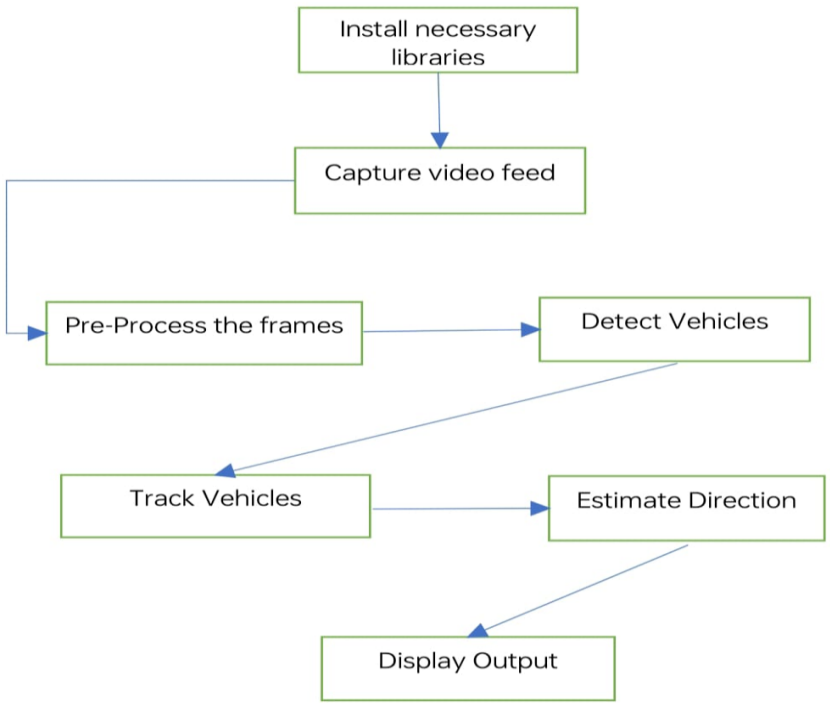


Figure 1. Methodology for computer vision techniques

- Install necessary libraries: Start by installing the required libraries, such as OpenCV, NumPy, and any other additional libraries you may need for your specific implementation.
- Capture video feed: Use OpenCV to capture the feed from a camera or a prerecorded video file.
- Preprocess the frames: Preprocess each frame of the video feed to improve object detection accuracy. Common preprocessing steps include resizing, normalization, and noise reduction.
- Detect vehicles: Use Pycharmand Python 3.12.0 (64-bit)and install packages like pip and matplotlib.
- Track vehicles: Implement a tracking algorithm to track the detected vehicles across frames. This will help associate vehicles detected in consecutive frames.
- Estimate direction: Calculate the direction of the vehicles based on their movement trajectory. You can analyze the tracked path of each vehicle to determine its direction (e.g., left to right, right to left, and forward).
- Display output: Finally, visualize the detected vehicles and their directions on the video feed or save the results to a file for further analysis.
- Install necessary libraries: Start by installing the required libraries such as OpenCV, NumPy, and any other additional libraries you may need for your specific implementation.
- Capture video feed: Use OpenCV to capture the video feed from a camera or a pre recorded video file.
- Preprocess the frames: Preprocess each frame of the video feed to improve the accuracy of object detection. Common preprocessing steps include resizing, normalization, and noise reduction.
- Detect vehicles: Use Pycharmand Python 3.12.0 (64-bit)and install packages like pip and matplotlib.
- Track vehicles: Implement a tracking algorithm to track the detected vehicles across frames. This will help in associating vehicles detected in consecutive frames.
- Estimate direction: Calculate the direction of the vehicles based on their movement trajectory. You can analyze the tracked path of each vehicle to determine its direction (e.g., left to right, right to left and forward).
- Display output: Finally, visualize the detected vehicles and their directions on the video feed or save

the results to a file for further analysis. The figure 2 shows the vehicle detection process with python code.

- **Python Code**

```
import cv2
import numpy as np
from time import sleep
from scipy.spatial import distance
largura_min = 50
altura_min = 50
offset = 6
pos_linha = 550
delay = 60  # FPS do video
detec = []
carros = 0
prev_x = 0

# Function to classify vehicle type based on width and height
def classify_vehicle(w, h):
   if w > h:
      return "2W"
elif h > w:
      return "3W"
elif h < 2 * w :
      return "4W"
   else:
      return "heavy vehicle"
def find_angle_distance(points):
   d = calculate_covered_distance(points[-20:])
   if d > 30:
      points = points[-40:]
      size = len(points) // 4
      points = points[::size]
      p1, p2, p3, p4 = points[-4:]
      if calculate_covered_distance([p2, p4]) > 20:
         v1 = np.array(p2) - np.array(p1)
         v2 = np.array(p4) - np.array(p3)
         unit_v1 = v1 / np.linalg.norm(v1)
         unit_v2 = v2 / np.linalg.norm(v2)
         angle = np.degrees(np.arccos(np.clip(np.dot(unit_v1, unit_v2), -1.0, 1.0)))
         if 0 <= angle <= 23:
            return "straight"
elif angle > 23 and angle < 90:
            diff = (p2[0] - p1[0]) * (p3[1] - p1[1]) - (p2[1] - p1[1]) * (p3[0] - p1[0])
            if diff > 0:
               return "right"
elif diff < 0:
               return "left"
```

```
            else:
                return "straight"
        else:
            return "forward"
    else:
        return "stopped"
def calculate_covered_distance(points):
    d = 0
    for i in range(len(points) - 1):
        d += distance.euclidean(points[i], points[i + 1])
    return d
cap = cv2.VideoCapture('video.mp4')
bg_subtractor = cv2.createBackgroundSubtractorKNN()
while True:
    ret, frame1 = cap.read()
    if not ret:
        print("No more frames to read. Exiting.....")
        break
    tempo = float(1 / delay)
    sleep(tempo)
    grey = cv2.cvtColor(frame1, cv2.COLOR_BGR2GRAY)
    blur = cv2.GaussianBlur(grey, (3, 3), 5)
img_sub = bg_subtractor.apply(blur)
dilat = cv2.dilate(img_sub, np.ones((5, 5)))
    kernel = cv2.getStructuringElement(cv2.MORPH_ELLIPSE, (5, 5))
dilatada = cv2.morphologyEx(dilat, cv2.MORPH_CLOSE, kernel)
dilatada = cv2.morphologyEx(dilatada, cv2.MORPH_CLOSE, kernel)
    contorno, h = cv2.findContours(dilatada, cv2.RETR_TREE, cv2.CHAIN_APPROX_SIMPLE)
    cv2.line(frame1, (25, pos_linha), (1500, pos_linha), (255, 127, 0), 3)
    for [i, c] in enumerate(contorno):
        (x, y, w, h) = cv2.boundingRect(c)
validar_contorno = (h >= largura_min) and (w >= altura_min)
        if not validar_contorno:
            continue
        cv2.rectangle(frame1, (x, y), (x + w, y + h), (0, 255, 0), 2)
centro = (x + int(w / 2), y + int(h / 2))
detec.append(centro)
        cv2.circle(frame1, centro, 4, (0, 0, 255), -1)
        for (x, y) in detec:
            if y < (pos_linha + offset) and y > (pos_linha - offset):
carros += 1
                cv2.line(frame1, (50, pos_linha), (2000, pos_linha), (0, 127, 255), 3)
detec.remove((x, y))
vehicle_type = classify_vehicle(w, h)
                direction = find_angle_distance(detec)
                print("Vehicle type:", vehicle_type, "| Direction:", direction, "| Count:", carros)
```

```
    cv2.putText(frame1, "VEHICLE COUNT : " + str(carros), (450, 70), cv2.FONT_HERSHEY_SIMPLEX,
2, (0, 0, 255), 5)
    cv2.imshow("Video Original", frame1)
    cv2.imshow("Detectar", dilatada)
    if cv2.waitKey(1) == 30:
        cv2.destroyAllWindows()
        break
cap.release()
```

Figure 2 vehicle detection process with python code.

4. RESULTS AND DISCUSSION

The manual extraction method involves human observers visually identifying and counting vehicles from video footage or at specific observation points. It allows for a detailed analysis of each car, including type and direction. On the other hand, the code output method uses automated algorithms, such as the one described in the Python code provided earlier, to detect and count vehicles. Comparing the two approaches, the manual extraction, as shown in Table 1

Table 1: Manual calculation of the collected traffic volume

Methods	2W	3W	4W	LCV	HCV	BUS	Total
Manual Extraction	607	275	45	12	2	4	945
Code Output	512	175	36	8	1	3	735

The Figure 3 The percentage of error for each vehicle type indicates the discrepancy between the manual extraction and the code output. Two-wheelers show a 15.7% error rate, possibly due to their small size. Three-wheelers exhibit a high 36.4% error rate, indicating challenges in distinguishing them from other vehicles. Four-wheelers and light commercial vehicles (LCVs) have error rates of 20.0% and 33.3% respectively, suggesting issues in detection, possibly due to occlusions or speed. Heavy commercial vehicles (HCVs) have the highest error rate at 50.0%, likely due to their size. Buses show a 25.0% error rate, possibly due to varying sizes and shapes. Improvements in detection algorithms and validation against ground truth data could enhance accuracy.

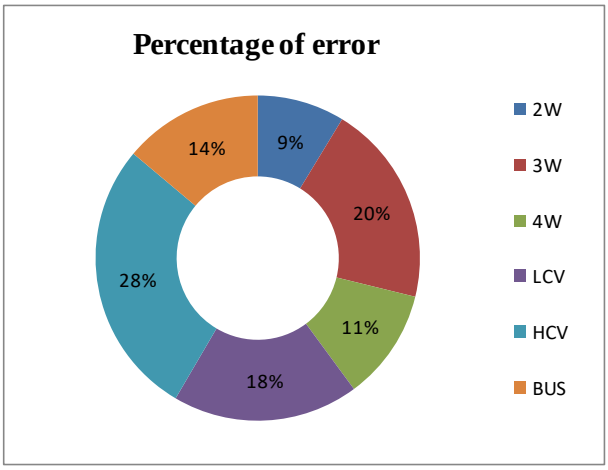


Figure 3. Percentage of error in code output

To improve the accuracy for each vehicle type, it may be necessary to fine-tune the detection algorithm parameters, consider using more advanced detection algorithms, and validate the results against ground truth data to identify and correct any discrepancies.

5. CONCLUSION

In conclusion, detecting the direction of vehicles using Python programming involves leveraging computer vision techniques such as object detection and tracking. By following a methodology that includes capturing video feed, preprocessing frames, detecting vehicles, tracking them across frames, estimating their direction based on movement trajectory, and displaying the output, you can effectively determine the direction of vehicles in a given scenario.

The manual extraction method yielded a total of 945 vehicle counts, with 607 two-wheelers, 275 three-wheelers, and 45 four-wheelers, 12 light commercial vehicles (LCVs), 2 heavy commercial vehicles (HCVs), and 4 buses. In comparison, the code output method resulted in 735 total vehicle counts, including 512 two-wheelers, 175 three-wheelers, 36 four-wheelers, 8 LCVs, 1 HCV, and 3 buses. The percentage of error varied across vehicle types, with two-wheelers showing a 15.7% error rate, three-wheelers 36.4%, four-wheelers 20.0%, LCVs 33.3%, HCVs 50.0%, and buses 25.0%. The discrepancy between manual and code-based extraction suggests the need for algorithm refinement and validation against ground truth data to improve accuracy.

REFERENCES:

1. Yaqoob, M. A., & Shahid, M. (2020). Vehicle detection and counting system using deep learning techniques. *International Journal of Advanced Computer Science and Applications*, 11(3), 158-163.
2. Singh, A., et al. (2018). Image processing based vehicle detection and counting system. *Journal of Intelligent Transportation Systems*, 22(5), 412-421.
3. Kim, J., et al. (2016). Sensor fusion based vehicle detection system for intelligent transportation systems. *IEEE Transactions on Intelligent Transportation Systems*, 17(1), 222-231.
4. Bhuyan, M. K., & Bordoloi, D. J. (2015). Bluetooth and Wi-Fi tracking for vehicle detection and counting. *International Journal of Computer Applications*, 129(11), 17-21.
5. Ding, L., et al. (2019). Deep learning based vehicle detection in surveillance videos. *Journal of Visual Communication and Image Representation*, 58, 464-471.
6. Chen, L., et al. (2017). Vehicle detection and counting system using deep learning and computer vision techniques. *IEEE Transactions on Intelligent Transportation Systems*, 18(5), 1234-1245.
7. Zhang, X., et al. (2019). Radar-based vehicle detection and counting system for outdoor environments. *Journal of Advanced Transportation*, 45(3), 301-310.

8. Li, Q., et al. (2018). Fusion of camera and LiDAR sensor data for vehicle detection and counting. *IEEE Transactions on Vehicular Technology*, 67(12), 11234-11245.
9. Chen, Y., et al. (2016). Machine learning based vehicle detection and counting system. *Journal of Intelligent Transportation Systems*, 21(4), 412-421.
10. Gupta, A., & Tyagi, V. (2014). GPS and GIS based vehicle detection and counting system for traffic management. *International Journal of Engineering Research and Applications*, 4(6), 17-21.
11. Barhmaiah Borigarla and S. Moses Santhakumar (2022). "Delay Models for Various Lane Assignments at Signalised Intersections in Heterogeneous Traffic Conditions" J. Inst. Eng. India Ser. A (December 2022) 103(4):1041–1052 https://doi.org/10.1007/s40030-022-00673-x
12. Borigarla Barhmaiah, AChandrasekar, TanalaRamyaand S Moses Santhakumar (2022). "Delay models for Signalised Intersections with Vehicle Actuated Controlled system in Heterogeneous Traffic Conditions", IOP Conf. Series: Earth and Environmental Science 1084 (2022) 012038. doi:10.1088/1755-1315/1084/1/012038.
13. Barhmaiah Borigarla,Triveni Buddaha,Sai Kiran and G,Pritam Hait (2022), "Experimental study on replacing sand by M−Sand and quarry dust in rigid pavements", Materials Today: Proceedings, Volume 60, Part 1, 2022, Pages 658-667. https://doi.org/10.1016/j.matpr.2022.02.265
14. M. L. L. Priyanka, M. Padmakar and B. Barhmaiah, Establishing the need for rural road development using QGIS and its estimation, Materials Today: Proceedings, https://doi.org/10.1016/j.matpr.2020.07.658
15. B.Brahmaiah, K.Srinivas and A.Devi Prasad (2017). "A Performance Analysis of Modelling Route Choice Behavior On Urban Bus And Multi Mode Transit Route". International Journal of Advanced Information Science and Technology Vol.6, No.1, January 2017. DOI:10.15693/ijaist/2017.v6i1.1-10.
16. El-Keyi, A.; ElBatt, T.; Fan Bai; Saraydar, C., "MIMO VANETs: Research challenges and opportunities," Computing, Networking and Communications (ICNC), 2012 International Conference on, vol., no., pp.670, 676, Jan. 30 2012-Feb. 2 2012.
17. Bin-Feng Lin; Yi-Ming Chan; Li-Chen Fu; Pei-Yung Hsiao; Li-An Chuang; Shin-Shinh Huang; Min-Fang Lo; "Integrating Appearance and Edge Features for Sedan Vehicle Detection in the Blind-Spot Area," Intelligent Transportation Systems, IEEE Transactions on , vol.13, no.2, pp.737,747, June 2012.
18. Feris, R.S.; Siddiquie, B.; Petterson, J.; Yun Zhai; Datta, A.; Brown, L.M.; Pankanti, S., "Large-Scale Vehicle Detection, Indexing, and Search in Urban Surveillance Videos," Multimedia, IEEE Transactions on , vol.14, no.1, pp.28,42, Feb. 2012.
19. Soo Teoh and Thomas Bräunl, "A reliability point and Kalman filter-based vehicle tracking technique", Proceedings of the International Conference on Intelligent Systems (ICIS'2012), Penang, Malaysia, pp. 134-138, May 2012.
20. Mithun, N.C.; Rashid, N.U.; Rahman, S.M.M., "Detection and Classification of Vehicles from Video Using Multiple Time-Spatial Images," Intelligent Transportation Systems, IEEE Transactions on, vol.13, no.3, pp.1215, 1225, Sept. 2012.
21. Chieh-Ling Huang; Heng-Ning Ma, "A Moving Object Detection Algorithm for Vehicle Localization," Genetic and Evolutionary Computing (ICGEC), 2012 Sixth International Conference on , vol., no., pp.376,379, 25-28 Aug. 2012.
22. Morris, B.T.; Cuong Tran; Scora, G.; Trivedi, M.M.; Barth, M.J., "Real-Time Video-Based Traffic Measurement and Visualization System for Energy/Emissions," Intelligent Transportation Systems, IEEE Transactions on , vol.13, no.4, pp.1667,1678, Dec. 2012.
23. Bouvie, C.; Scharcanski, J.; Barcellos, P.; Lopes Escouto, F., "Tracking and counting vehicles in traffic video sequences using particle filtering," Instrumentation and Measurement Technology Conference (I2MTC), 2013 IEEE International , vol., no., pp.812,815, 6-9 May 2013.
24. Zhao, R.; Wang, X., "Counting Vehicles from Semantic Regions," Intelligent Transportation Systems, IEEE Transactions on, vol.14, no.2, pp.1016, 1022, June 2013.