IJCRT.ORG ISSN : 2320-2882

INTERNATIONAL JOURNAL OF CREATIVE RESEARCH THOUGHTS (IJCRT)

An International Open Access, Peer-reviewed, Refereed Journal

Engine Recognizable proof and Counting Strategy Based on Computerized Picture Handling in Python

Rani Ingle
Mtech in Department of Computer Science
G H Raisoni University,
Amravati, India

***Summary**:The vehicle counting process provides good information about traffic flow, vehicle accident occurrences, and peak traffic times on the road. An acceptable technique for achieving these goals is the use of digital image processing techniques in the video output of roadside cameras. This paper introduces a vehicle counter classifier based on a combination of different video image processing methods such as object detection, edge detection, frame difference, and Kalman filter. The implementation of the proposed technique was performed using the Python programming language. This white paper describes the methods used for image processing for traffic flow counting and classification using various libraries and algorithms with real-time images.*

INTRODUCTION

Summary: The vehicle counting process provides good information about traffic flow, vehicle accident occurrences, and peak traffic times on the road. An acceptable technique for achieving these goals is the use of digital image processing techniques in the video output of roadside cameras. This paper introduces a vehicle counter classifier based on a combination of different video image processing methods such as object detection, edge detection, frame difference, and Kalman filter. The implementation of the proposed technique was performed using the Python programming language. This white paper describes the methods used for image processing for traffic flow counting and classification using various libraries and algorithms with real-time images.

However, image processing is time consuming and needs automation to save time counting and classifying images. In the current era of Python-type programming languages, significant time savings have been added for image processing and vehicle detection, counting, and classification. This paper provides information on the path to image processing, the types of filters used, and the techniques proposed to accurately detect, count, and classify images.

BACKGROUND INFORMATION

Video Processing

Video processing is a subcategory of digital signal processing technology where the input and output signals are video streams. For computers, one of the best ways to reach your video analysis goals is to use image processing methods at each video frame. In this case, the movement is achieved simply by comparing successive frames

Video processing includes a pre-filter that can provide contrast changes and noise reduction along with pixel size conversion of video images. Highlighting specific areas of the video, removing improper lighting effects, eliminating camera movements, and removing edge artifacts are all feasible with video processing methods. Python's OpenCv library is loaded with functions that allow you to work with videos and images. OpenCV Python uses Numpy, a library for math operations that uses MATLAB-style syntax. All OpenCV array structures are converted to and from numpy arrays. This also facilitates integration with other libraries that use Numpy, such as SciPy and Matplotlib.

RGB to Grayscale Conversion:

In video analysis, the conversion of RGB color images to grayscale mode is done by image processing methods. The main purpose of this conversion is to process grayscale images with more acceptable results compared to the original RGB image. Video processing techniques require the sequence of captured video frames to be converted from RGB color mode to a gray level from 0 to 255. When converting an RGB image to grayscale mode, you need to get the RGB value for each pixel and prepare a single value as output that reflects the percentage of brightness for that pixel.

Power-Law Transformation:

Emphasis on an image gives you better contrast and more detail than an unenhanced image. There are several image enhancement techniques such as power law transformation, linear programming, and logarithmic programming. Image enhancement can be performed by one of these grayscale transformations. Among them, the power law conversion method is a suitable method, and its basic form is as follows.

$V = A v^{\gamma}$

If V and v are the gray levels of the outputs and inputs, then γ is the gamma value and A is a positive constant (generally if A = 1). The Python code that implements power law conversion is-
power_law_transformation = cv2.pow (grey, 0.6)

The second argument is the gamma value. Therefore, choosing the right value for γ plays an important role in the image enhancement process and can provide the right details that can be identified in the image.

Canny Edge Detection:

Object detection can be performed using the image matching feature and edge detection. Edges are the points of a digital image where the brightness and height of the gray level of the image suddenly change. The main task of edge detection is to find all the pixels in the image that correspond to the edges of the object displayed in the image. Among the various edge detection methods, the Canny algorithm is a simple and powerful edge detection method. Edge detection is prone to noise in the image, so the first step is to use a 5x5 Gaussian filter to remove the noise in the image. The smoothed image is then filtered using the Sobel kernel both horizontally and vertically to obtain first-order derivatives in the horizontal (Gx) and vertical (Gy) directions. From these two images, you can find the edge gradient and orientation for each pixel as follows:

Edge gradient (G) = √ G2x + G2y (2)
Angle (θ) = tan-1 (Gy / Gx) (3)

The direction of the gradient is always perpendicular to the edge. It is rounded to one of four angles that represent the vertical, horizontal, and two diagonal directions. After the gradient size and orientation are determined, a full scan of the image is performed to remove unwanted pixels that may not form edges. For this purpose, each pixel is checked to see if it is a maximum near the gradient direction. OpenCV wraps all of the above in one function cv2.Canny.

The Kalman Filter:

Images usually have a lot of speckles caused by noise and should be removed by filtering. The Kalman filter is a powerful and useful tool for estimating a particular process using some kind of feedback information [14]. The Kalman filter is used to provide an improved estimate based on a series of noisy estimates. This filter shows that the basic process needs to be modeled with a linear dynamic structure.

xk = Fk-1xk-1 + wk-1 (4)
yk = Hkxk + vk (5)

where xk and yk are state and measurement vectors, wk and vk are process and measurement noise, Fk and Hk are transition and measurement values, and k is the desired time step. The Kalman filter also shows that the measurements and error terms represent a Gaussian distribution. That is, in vehicle detection, each vehicle can only be tracked with one Kalman filter. Therefore, the number of Kalman filters applied to each video frame depends on the number of vehicles detected.

PREVIOUS WORKS

The use of image / video processing and object recognition methods for vehicle detection and traffic flow estimation has received a great deal of attention for several years. The vehicle detection / tracking process was performed using one of the following methods.

- Adjustment
- Thresholds and segmentation
- Point detection
- Edge detection
- Frame identification
- Optical flow method

It can be said that one of the most important studies in the field of object recognition that led to the Autoscope video recognition system is shown in. Some works, such as, use the forward and backward image diff method to extract moving vehicles in the road view. Some studies, such as have shown that using feature vectors from the image area for vehicle detection targets can be very efficient. Some have provided accurate estimates of vehicle dimensions using a set of coordinate mapping functions, as seen in addition, some studies have developed various object detection boosting algorithms using machine learning techniques that can detect and classify moving objects by both type and color. The named approach has both advantages and disadvantages.

PROPOSED TECHNIQUE

Unlike previous work, the method proposed in this article uses a combination of frame differentiation and edge detection algorithms to improve the quality and accuracy of vehicle detection. By using the Kalman filter, the position of each vehicle is correctly estimated and tracked. This filter is also used to classify detected vehicles into different fixed groups and count them individually to provide useful information for traffic flow analysis. A flow chart of this method is shown in Figure 1.

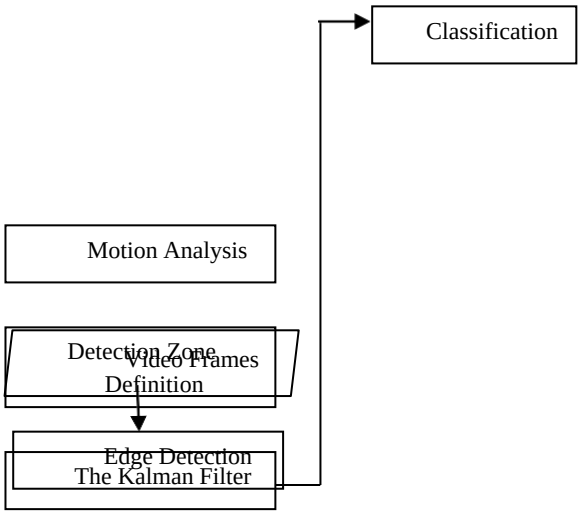

Fig. 1. Flowchart of the Technique

Based on Figure 1, this technique involves the following steps: Image enhancement process, edge detection, motion analysis using a combination of different techniques, detection zone definition, Kalman filter, vehicle classification and counting. It must be said that some assumptions were made in this work.

- No sudden changes in direction are expected
- No car accidents or accidents expected
- Vehicles have physical and legal restrictions
- Motion scenes are recorded with the road surface viewed from above

The proposed procedure to distinguish and check vehicles is displayed as underneath:

Grayscale Image Generation and Image Enhancement:

To get better results, vehicle detection process should be performed in the grayscale image domain. Hence a RGB to grayscale conversion is performed on each video frame. To achieve an appropriate threshold level and make results more suitable than the input image, each frame should be brought in contrast to background. Among several grayscale transformations, power-law method has been used in this work. For color conversion we use the function cv2.cvtColor(input_image,flag) where flag determines the type of conversion. Use the cv2.COLOR_BGR2GRAY flag to convert to grayscale. Experimental results in various situations show that the best results are obtained when the γ value is set to 0.6, as shown in Figure 2. This figure shows the result of applying various γ values to a grayscale converted image. Here, Section A is the input RGB color frame, and B and C are grayscale versions with gamma values of 0.6 and 0.9, respectively. The implementation of the result in Figure 2 can be obtained using the Python code shown in Figure 3.

Fig. 2. Input RGB Video Frame (A) and Grayscale Converted With Different Γ Values (B and C)

```
while True:

    ret, frame = cap.read()

    if ret == True:

        gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
        gray =gray/255.0
        power_law_transformation=cv2.pow(gray,0.6)
        cv2.imshow('tranformed',power_law_transformation)
```

Fig. 3. Code for Conversion from RGB to Grayscale and Image Enhancement

Edge Detection:

To achieve detection aims, each picture (video frame) comprises three key features. Edges, curves, and points are examples of these features. Among the attributes stated, edge pixels are a good choice. We can detect edge pixels, which are the major features of passing automobiles in a roadway video frame, by processing picture pixels. The Canny operator, which has been employed in this study, is one of the most used approaches to detect the edges of a picture. Figure 4 shows the result, while Figure 5 shows the code that goes with it. The output of the edge detection procedure is shown in a binary picture (threshold) with the identified edge pixels, as can be seen.

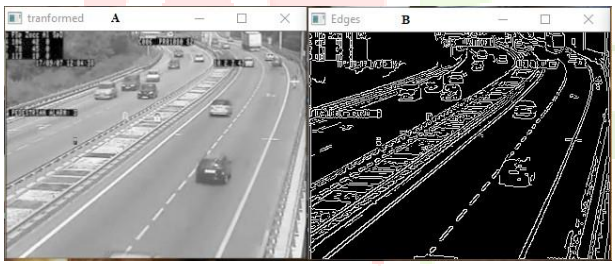

Fig. 4. A: Original Image B: Edge Detection Result

```
hsv=cv2.cvtColor(frame,cv2.COLOR_BGR2HSV)
lower_red=np.array([30,150,50])
upper_red=np.array([255,255,180])
mask=cv2.inRange(hsv,lower_red,upper_red)
res=cv2.bitwise_and(frame,frame,mask=mask)
edges=cv2.Canny(frame,100,200)
cv2.imshow('Edges',edges)
```

Fig. 5. Code for Canny Edge Detection

The following step is to extract moving edges from sequential video frames and analyse the edge data to produce quantitative geometric measurements of passing cars.

Background Subtraction:

The static parts of the sequential video frame should be cleaned up using the provided thresholds. The main challenge here is that the performance of image analysis algorithms can be degraded under darkness, glare, long shadows, or poor lighting at night, leading to excessive noise. Therefore, in these situations, the grayscale image may not be specified, which complicates the recognition task a bit. Edges basically separate two different areas: static areas (roads) and dynamic areas (moving vehicles). Then erase the static background to find the moving object in each frame. The result zone leaves only the vehicle and some details as moving objects within a continuous frame that change from frame to frame. This task used a combination of forward and backward frame diffs and Sobel edge detectors. With this method, three consecutive frames are selected and the middle frame is compared with the previous and next frames. Therefore, we will use the extracted edges of each frame detected by the cany edge detection obtained in the previous section. You can then get the frame difference by subtracting two consecutive pairs of generated binary images, as in Equation 6.

Where Fn-1 is previous frame, Fn is current frame and Fn+1 is the next frame. This process continues to the last three sequential video frames. The output result is demonstrated in Figure 6. The python code is represented in Figure 7. In this figure A, B and C represent three sequential frames, where D demonstrates the output background subtraction method. Using this technique moving vehicles are detected in three sequential frames.

Fig. 6. Proposed Moving Vehicle Detection Technique and Background Subtraction (A,B,C and D)

Fig. 7. Code for Background Subtraction

Detection Zone:

As an observation (detection) zone, a region should be defined to display moving vehicle' s edges in a bounding box at the time that the vehicle enters it. This zone is in the middle of the screen and covers 1/3 of its height and 3/5 of its width (considering minimum and maximum available size of detectable passing vehicles in pixels). This area which contains the most traffic can embed both small and long vehicles and the main goal of defining it is to avoid perspective challenges and wrong type counts. Based on proposed method in background subtraction level, a vehicle is detected in three sequential frames. When a moving vehicle is detected, a bounding box whelming vehicle borders in binary image is drawn.

The Kalman Filter:

The bounding box can also be used to count and classify passing vehicles. This can be done with Kalman filter technology. In road video, the edge detection feature provides an incorrect position for the moving vehicle, but you need to improve your knowledge of the vehicle's current position. Since we cannot guarantee a perfect measurement due to the movement of the object, we need to filter the

measurement to get the best estimate of for the exacttrace. The Kalman filter can also optimally estimate the current position of each vehicle and predict the position of the vehicle in future video frames by minimizing noise interference. It is also used to stop tracking vehicles driving in the opposite direction in street video. Edge detection can find moving objects, but the Kalman filter makes optimal position estimates based on a series of position measurements.

The linear Kalman filter is simpler and used in proposed technique. Consider parameter A as area of vehicle' s bounding box, which has been detected in frame differentiation phase and p(x, y) is the center point of the vehicle where x and y are its distances from horizontal and vertical edges. Now by integration of proposed parameter in (7) and (8) equations resulted in the following vectors.

xk = [x, y, A, vx, vy, vA] T (7) yk = [x, y, A] T (8)

Where vA is the rate of changes in vehicle' s bounding box, vx and vy are the speed of changes in the movement of vehicle' s center point. Subsequently using the Kalman filtering technique, the position of each vehicle can be estimated and tracked better. Finally an identifier is allocated to each passing vehicle for counting and classification purposes.

The Kalman Filter is a unsupervised algorithm for tracking a single object in a continuous state space. Given a sequence of noisy measurements, the Kalman Filter is able to recover the " true state" of the underling object being tracked. It is implemented using the pykalman library of python.

Sample code-

```
from pykalman import KalmanFilter

kf = KalmanFilter(initial_state_mean=0, n_dim_obs=2)
```

The traditional Kalman Filter assumes that model parameters are known beforehand. The KalmanFilter class however can learn parameters using KalmanFilter.em() (fitting is optional). Then the hidden sequence of states can be predicted using KalmanFilter.smooth():

```
measurements = [[1,0], [0,0], [0,1]]

kf.em(measurements).smooth([[2,0], [2,1], [2,2]])[0]


array([[ 0.85819709],

[ 1.77811829],

[ 2.19537816]])
```

Common uses for the Kalman Filter include radar and sonar tracking and state estimation in robotics. This module implements two algorithms for tracking: the Kalman Filter and Kalman Smoother. In addition, model parameters which are traditionally specified by hand can also be learned by the implemented EM algorithm without any labeled training data. All three algorithms are contained in the KalmanFilter class in this module.

Counting and Classification Functions:

Vehicle counters are used in computing capacity, establishing structural design criteria and computing expected roadway user revenue. Typically in proposed technique vehicles are classified as four common types:

- Type1: bicycles, motorcycles
- Type2: motorcars
- Type3: pickups, minibuses
- Type4: buses, trucks, trailers

It is necessary to have the width and length of each vehicle' s bounding boxes in pixels to diagnose that the passing vehicles belongs to which of the mentioned types. The area of each bounding box shows that which type should be allocated for the vehicle. Each vehicle type can be shown by a special rectangle color. Type 1 has been represented by red, where Type2, Type 3 and Type 4 have been characterized by green, blue and yellow rectangles, respectively.

In counting step, four isolated counters used for each vehicle type and a total counter is needed to store the sum value of them. All counters should count just the vehicles which are passing in a specific direction. So if a vehicle stops, turns or moves in wrong direction in the detection zone, it should not be counted. In this technique, counting is according to the number of moving vehicles detected in the detection zone and classified in one of mentioned groups.

Total passed vehicles, which will be shown in yellow, help to analyze traffic flow in a period of time. Also by calculating the bounding boxes height and width in pixels, vehicle types can be distinguished and counted by related counters. Furthermore, in both counted vehicles, edges will be covered with green rectangles, which shows that they belong to Type 2 (even the green numbers inside bounding boxes confirm this result).

CONCLUSION

A methodology based on python language programming is proposed in this research. Python has many useful libraries, such as numpy, matplotlib, and scipy, that can assist engineers count traffic, classify traffic, and save time. Traffic flow is essential information for transportation planning, and getting it right and processing it in a timely manner is a difficult task for transportation and highway engineers. From the standpoint of road construction design and traffic planning, these technologies will be quite valuable.

REFERENCES

[1] D. Beymer, P. McLauchlan, B. Coifman, J. Malik, "A Real-time Computer Vision System for Measuring Traffic Parameters," IEEE Conference on Computer Vision and Pattern Recognition, 1997.

[2] M. Fathy, M. Y. Siyal, "An Image Detection Technique, Based on Morphological Edge Detection and Background Differencing for Real-time Traffic Analysis," Pattern Recognition Letters, Vol. 16, pp. 1321-1330, 1995.

[3] V. Kastrinaki, M. Zervakis, K. Kalaitzakis, "A Survey of Video Processing Techniques for Traffic Applications," Image and Vision Computing, Vol. 21, pp. 359-381, 2003.

[4] D. A. Forsyth, J. Ponce, "Computer Vision: A Modern Approach," Prentice Hall, 2003.

[5] T. R. Currin, "Turning Movement Counts. In Introduction to Traffic Engineering: A Manualfor Data Collection and Analysis," Stamford WadsworthGroup, pp. 13"23, 2001.

[6] W. Yao, J. Ostermann, Y. Q. Zhang, "Video Processing and Communications," Signal Processing Series, ISBN: 0-13-017547-1, Prentice Hall, 2002.

[7] P. Choudekar, S. Banerjee, M. K. Muju, "Real Time Traffic Light Control Using Image Processing," Indian Journal of Computer Science andEngineering, Vol. 2, No. 1, ISSN: 0976-5166.

[8] N. Chintalacheruvu, V. Muthukumar, "Video Based Vehicle Detection and Its Application in Intelligent Transportation Systems," Journal of Transportation Technologies, Vol. 2, pp. 305-314, 2012.

[9] R. Milances Gil, S. Pun, T. Pun, "Comparing Features for Target Tracking in Traffic Scenes," Pattern Recogition, Vol. 29, No. 8, pp. 1285-1296, 1996.

[10] E. Atkociunas, R. Blake, A. Juozapavicius, M. Kazimianec, ?Image Processing in Road Traffic Analysis,? Nonlinear Analysis: Modelling and Control, Vol. 10, No. 4, pp. 315?332, 2005.

[11] X. Fu, Z. Wang, D. Liang, J. Jiang, ?The Extraction of Moving Object in Real-Time Web-Based Video Sequence,? 8th International Conference on Digital Object Identifier, Vol. 1, pp. 187-190, 2004.

[12] V. Khorramshahi, A. Behrad, N. K. Kanhere, ?Over- Height Vehicle Detection in Low Headroom Roads Using Digital Video Processing,? World Academy of Science, Engineering and Technology, 2008.

[13] J. Zhou, D. Gao, D. Zhang, ?Moving Vehicle Detection for Automatic Traffic Monitoring,? IEEE Transactions on Vehicular Technology, Vol. 56, NO.1, 2007.

[14] G. Welch, G. Bishop, ?An Introduction to the Kalman Filter?, The University of North Carolina at Chapel Hill, 2006.

[15] P. G. Michalopoulos, ?Vehicle Detection Video Through Image Processing: The Autoscope System,? IEEE Transactions on Vehicular Technology, Vol. 40, No. 1, 1991.

[16] A. H. S. Lai, G. S. K. Fung, N. H. C. Yung, ?Vehicle Type Classification from Visual-Based Dimension Estimation,? IEEE Intelligent Transportation Systems Conference, pp. 201-206, 2001.

[17] D. G. Lowe, ?Distinctive Image Features from Scaled-Invariant Keypoints,? International Journal of Computer Vision, pp. 91-110, 2004.

[18] P. T. Martin, G. Dharmavaram, A. Stevanovic, ?Evaluation of UDOT?s Video Detection Systems: System?s Performance in Various Test Conditions,? Report No: UT-04.14, 2004.

[19] O. Hasegawa, T. Kanade, ?Type Classification, Color Estimation, and Specific Target Detection of Moving Targets on Public Streets,? Machine Vision and Applications, Vol. 16, No. 2, pp. 116-121, 2005.

[20] R. Cucchiara, M. Piccardi, P. Mello, ?Image Analysis and Rule-based Reasoning for a TrafficMonitoring System,? IEEE Transactions on Intelligent Transportation Systems, Vol. 1, Issue 2, pp 119-130, 2000.

[21] Q. Cai, A. Mitiche, J. K. Aggarwal, ?Tracking Human Motion in an Indoor Environment,? International Conference on Image Processing, USA, Vol. 1, pp. 215-218, 1995.

[22] T. Le, M. Combs, Q. Yang, ?Vehicle Tracking basedon Kalman Filter Algorithm,? Technical Reports Published by the MSU Department of Computer Science, ID: MSU-CS-2013-02, 2013.

[23] Sh. Agarwal, A. Awan, D. Roth, ?Learning to DetectObjects in Images via a Sparse, Part-based Representation,? IEEE Transactions on Pattern Analysis and Machine Intelligence, 2004.

[24] Sh. Agarwal, D. Roth, ?Learning a Sparse Representation for Object Detection,? European Conference on Computer Vision, Vol. 1, ISBN: 978-3-540-43748-2, pp. 113-127, 2002.

[25] A. Torralba, K. P. Murphy, W. T. Freeman, ?Shared Features for Multiclass Object Detection,? Toward Category-Level Object Recognition, ISBN: 978-3- 540-68794-8, pp. 345-361, 2006.

[26] L. Bohang, L. Qingbing, C. Duiyong, S. Hailong, ?Pattern Recognition of Vehicle Types and Reliability Analysis of Pneumatic Tube Test Data under Mixed Traffic Condition,? 2nd International Asia Conference on Informatics in Control, Automation and Robotics, ISSN: 1948-3414, pp. 44-47, 2010.

[27] L. Feng, W. Liu, B. Chen, ?Driving Pattern Recognition for Adaptive Hybrid Vehicle Control,? SAE 2012 World Congress and Exhibition, pp. 169- 179, 2012.

[28] D. Simon, ?Optimal State Estimation: Kalman, H Infinity, and Nonlinear Approaches,? Wiley- Interscience, 2006.

[29] R. Gonzalez, R. E. Woods, ?Digital Image Processing,? 2nd Edition, Prentice-Hall, 2002.

[30] S. Siang Teoh, T. Bräunl, ?A Reliability Point and Kalman Filterbased Vehicle Tracking Technique,? International Conference on Intelligent Systems, pp. 134-138, 2012.

[31] K. Markus, ?Using the Kalman Filter to Track Human Interactive Motion Modeling and Initialization of the Kalman Filter for Translational Motion,? Technical Report, University of Dortmund,Germany, 1997.

[32] D. Nagamalai, E. Renault, M. Dhanuskodi. ?Implementation of LabVIEW Based Intelligent System for Speed Violated Vehicle Detection?, First International Conference on Digital Image Processing and Pattern Recognition, ISSN: 1865-0929, pp. 23-33, 2011.

[33] I. E. Igbinosa, ?Comparison of Edge Detection Technique in Image Processing Techniques?, International Journal of Information Technology and Electrical Engineering, ISSN: 2306-708X, Vol. 2,Issue 1, 2013.

[34] Learning OpenCV: Computer Vision with the OpenCV Library By Gary Bradski, Adrian Kaehler.