1. VEHICLE DETECTION FOR TRAFFIC FLOW ANALYSIS

V. Varsani, S. Ramalingam
School of Engineering & Technology, University of Hertfordshire
College Lane Campus, Hatfield, Hertfordshire AL10 9AB, UK

ABSTRACT

This paper looks at some of the algorithms that can be used for effective detection and tracking of vehicles, in particular for statistical analysis. The main methods for tracking discussed and implemented are blob analysis, optical flow and foreground detection. A further analysis is also done testing two of the techniques using a number of video sequences that include different levels of difficulties.
Keywords: Video analysis, vehicle detection, tracking, blob analysis, optical flow, background extraction, foreground extraction.

2. INTRODUCTION

As technology advances and high quality video cameras become inexpensive, there has been a rise in automated video analysis through video processing in recent years. The increase in the processing power of computers in conjunction with their memory has led to a growth of automated systems for video analysis that comprise of object tracking, which is an important field in computer vision. There has also been increased research in object tracking systems for military use. Combination of both these methods has led to numerous applications such as surveillance, teleconferencing, traffic monitoring, to name a few.

The key goal of traffic surveillance systems is the evaluation of traffic conditions determined by parameters such as traffic flow rate, average traffic speed, length of queue and traffic density. Typically, such methods employ vehicle detection followed by tracking techniques which need to be accurate and reliable[1-2]. Such systems detect objects entering the scene and start to track them throughout the video or up to a vanishing point after which the object becomes insignificant for tracking because the size of the object is small and is difficult to be differentiated from noise. The first few frames are used for background model estimation especially without any moving objects in the scene. Thus the background model collects the statistics of the background scene including objects such as road, trees, building, lamp posts, etc. making it possible to distinguish vehicles that would appear as foreground objects.

Detecting an object of interest comprises of two main steps namely video processing and object tracking. Video processing is a method of using filters to edit the video sequence, so the useful information can be extracted from the raw image scene. The process involves different techniques for filtering to obtain this useful information as each situation is unique and the solution to acquire this data differs from one video to another. Video processing is knowledge based in terms of the neighborhood distribution and filtering process to remove noise and mis-detections. Resulting objects are tracked until the duration of interest.

Object tracking is the process of automatically locating an object from one frame to another in a video sequence or in an image. There are however some problems involved with tracking objects, such as; difficult object shapes that need to be tracked, objects moving randomly, noise in the video that needs to be filtered, effect of long shadows, static objects in scenes affecting shape of moving objects, etc. For robust tracking, it is necessary to deal with shadows at motion detection level and at tracking level. Tracking methodologies vary depending on the applications for which it is implemented.

This paper presents an automated system for traffic analysis with moving vehicles on the road. A combination of foreground detection and blob analysis, and a combination of optical flow and blob analysis are both analyzed for performance evaluation. These methods are designed to overcome specific problems associated with the individual methodologies. The system is able to count the number of vehicles passing by in bidirectional lanes. Real-time processing is an essential requirement for usage in traffic surveillance applications and this paper achieves this performance when tested on video sequences. The system involves analyzing a sequence of road images which represents a flow of traffic observed at a given time and place.

The rest of the paper is organized as follows; Section II details the literature review highlighting some of the advantages and pitfalls with specific techniques. Section III specifies the algorithmic implementation for the two techniques namely Blob Analysis and Optical Flow. Section IV includes the testing for both the algorithms and the counter mechanism implemented for counting the number of vehicles in the scene. Section V concludes the report and includes any further development that can be done to this work.

3. LITERATURE REVIEW

Video processing and object tracking is a method that involves the use of filtering of video to acquire valuable information and use this information for detecting an object of importance and tracking it. There has been a great deal of research and ideas developed on object tracking. At present, several algorithms have been developed for the process of detection, a number of which involve an individual algorithm while others use a combination of two or more. Each of these algorithms has its advantages and drawbacks, as it can be seen that they are used to solve a particular problem of tracking. In this paper, we are concerned with two main tracking methods namely Blob Analysis and Optical Flow which are reviewed in this Section.

2.1 Blob Analysis Based Tracking Techniques

Blob analysis or blob tracking[1-4] is a method that is used to detect and track objects in which a blob is considered as an area of associated pixels. Blob Analysis typically involves the following steps: i) Background extraction, ii) Blob detection, iii) Blob analysis, iv) Blob tracking and v) Vehicle counting. Salvi uses the k-means clustering technique for blob detection[1] based on input information on a set of candidate blobs and their co-ordinate positions in the scene. The algorithm extracts only the relevant blobs in the scene using k-means clustering that estimates the centers of clusters and classifies the blobs to specific clusters based on distance measures. A contour detection algorithm finds the contours surrounding each class and verified by its size to reduce any false detections.

X. Song and R. Nevatia[2] used an algorithm for tracking vehicles for which the background model is known a-priori. A video sequence from cameras mounted on street poles is used. Their technique involved detecting vehicles by calculating the motion of the foreground blobs, by comparison to the learnt model of the background and then tracking using connections between the detected blobs. Tracking these blobs is achieved on the basis that the algorithm has already learnt the scene from the previous frames. The subtracted background is updated in changing light conditions. Nevertheless there are difficulties faced in doing this; change in background caused by camera shakes, moving objects in the foreground other than vehicles, blobs combining where the vehicles are close, etc. To get rid of the noise due to the pixels not registering as background pixel, morphological filtering is used. Problems are however faced even after the filtering is done; blobs merge together, blobs split or unwanted objects are tracked in the scene.

Thou-Ho Chen[4] used a rectangular patch as a model for vehicle classification. Blob analysis is used to determine steady features of moving objects such as extent of disperse, aspect ratio and area ratio and tracking is done by correlating these features in successive frames. The velocity of vehicle movement is also achieved through blob analysis.

A connected component based blob analysis is proposed in[5] to extract foreground objects using background subtraction. Further the objects of interest are segmented by comparing features of blobs between frames. Obtained motion vectors are grouped into clusters for future blob classification. The method has been demonstrated for tracking vehicles on the highway.

2.2 Optical Flow Based Tracking Techniques

Optical flow techniques approximate image motion based on local derivatives from a sequence of images thereby specifying how much each image pixel has moved between adjacent images. Optical flow techniques can be categorized as i) gradient based ii) correlation based iii) feature based, and iv) multigrid methods. Gradient based techniques perform a pixel by pixel matching between frames. The optical flow field requires magnitude and direction. Hence, in most cases, intensity variation alone is not sufficient. Smoothness constraints facilitate the estimation of optical flow[3]. Thus, correlation based techniques determine the maximum shift around each pixel that maximizes the correlation of gray levels between consecutive frames. Feature based techniques cluster pixels into blobs and thereby performs matching of blobs between frames. For each blob, the centroid is determined, which are estimated and matched during each frame. The accuracy of all of these techniques is affected by sensor noise namely that of derivative calculation. However, they are very useful techniques that perform better than human beings. Similar to the feature based methods, multi-grid methods use pixels with similar intensities to be grouped into labelled objects.

Nejadasi et. al[6], have proposed a gradient-based optical flow method to track vehicles based on both pixel and region based blobs. The system is made sensitive to small movements even under low contrast conditions as it considers both spatial and temporal changes simultaneously. Gern et.al[7] proposed the horizontal optical flow based technique to determine lane markings on road and learn the relative position of a vehicle with respect to these lane markings. The system is aimed at working even under worse weather conditions where the road markings are not clear. A correlation based approach is followed to determine correspondence of prominent structures between frames. A reference model under good weather conditions is first established to determine any deviation of movement of vehicles under bad weather conditions.

The two categories for vehicle detection namely Blob Analysis and Optical Flow are useful in that blob analysis is essential when segmented objects need to be properly classified as objects of interest in the moving scene. Objects in the moving scene can be easily determined by Optical Flow analysis even though there are computational errors involved. Hence, this paper analyses these two techniques to understand their strengths and weaknesses better.

4. VEHICLE DETECTION IMPLEMENTATION

A set of algorithms for vehicle detection and tracking is developed using MATLAB[8]. The two algorithms are referred to as:

Algorithm (1)	Foreground detection and blob analysis.(FDBA)
Algorithm (2)	Optical flow and blob analysis. (OFBA)

It is to be noted that in this paper, a comparison of the above algorithms with that of optical flow alone is not conducted. Hence, this paper compares the performance of Blob Analysis denoted by FDBA Vs Combined algorithm incorporating both Optical Flow and Blob Analysis and denoted by OFBA. The flowchart of the proposed system design is shown in Fig.1.

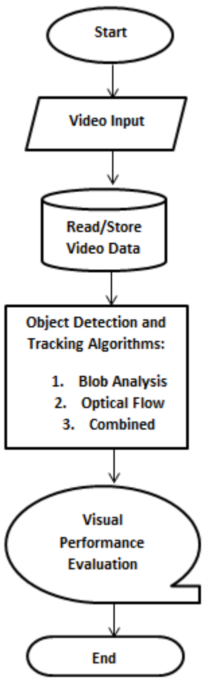


Figure 1. Flowchart of the proposed system.

3.1 Algorithm 1 - Foreground detection and blob analysis

Blob analysis uses the notion that the foreground keeps on changing relatively compared to the background. Using this concept, it subtracts the background from the foreground, and links these pixels that are moving and close to each other in the foreground as one blob.

To begin with the algorithm subtracts the background using the MATLAB inbuilt vision object **vision.ForegroundDetector**. A Gaussian mixture consisting of parameters such as the number of frames to train the background model, learning rate, and the minimum background ratio are tuned and determined to achieve the desired outcome. These steps are applied to all the frames of the video sequence using the MATLAB **Step** function, and a morphological filter is used and adjusted to smooth out the subtraction output. Then the filtered frames are forwarded for blob analysis, where different parameters are altered for the system object, some of these parameters include; bounding box output; which is used to return coordinates of the bounding box for tracking later on, and minimum blob area; the property specifies the minimum pixel area that can be regarded as a blob. At the end, a bounding box is inserted around the detected blobs in each frame and the video output is given. A frame of the video when the vehicles are tracked is shown with some discussions and problems faced by the algorithm. The results of these operations are as shown in Figs.2-4. (MATLAB examples)

From the figures it can be seen how the morphological filter works, Fig.2 shows the background subtraction without filtering, and Fig.3 shows the subtraction with morphological operators for filtering. It is seen that the image is smoother and sharper after filtering. Finally Fig.4 image shows the tracking in progress. Analysing Figs. 2 and 3 it can be concluded that the algorithm does not work as expected, it is seen that some neighbouring foreground pixels are not being detected as blobs hence they are not tracked. Examining the whole video, problems were also observed with the tracking; one of these problems seems to be that the blobs are not regarding the street lights as part of the background model which the road belongs to. This in effect does not track the vehicles when it passes by the street light. Another problem is that all the pixel area of the vehicle is not considered as a single blob and therefore when tracking the bounding box breaks up in to smaller boxes. For these reasons other techniques are looked at that do not have the same problem.

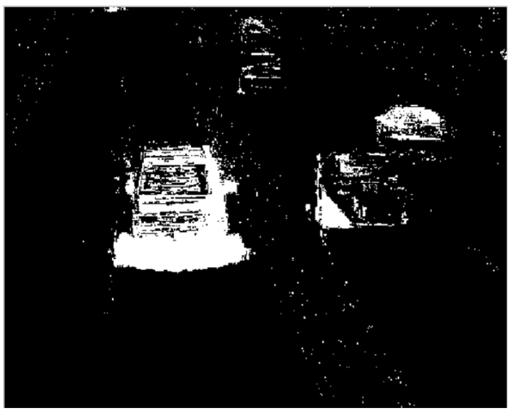


Figure 2. Algorithm1: Background subtracted frame.

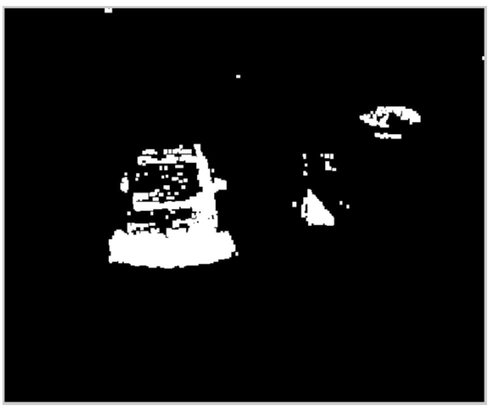


Figure 3. Algorithm1: Filtered background subtracted frame using morphological operators.

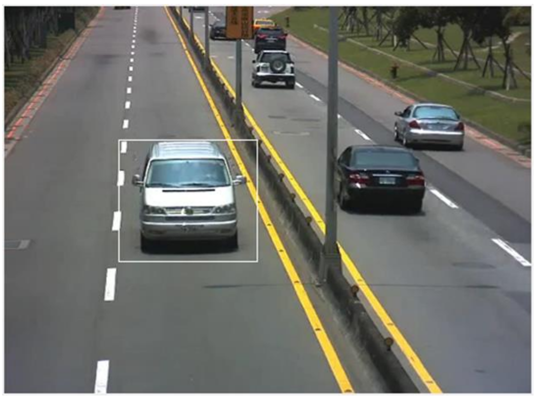


Figure 4. Algorithm1-FDBA: Video frame for the vehicles being tracked. (input for Figs2-3)

3.2 Optical flow

The optical flow technique is regarded as one of the most efficient way of detection and tracking when it comes to still backgrounds. However this is not the case when it comes to a moving background, as the algorithm works by calculating the movement of apparent velocity patterns of brightness patterns between successive frames. At first, the video file is read into MATLAB using the **vision.VideoFileReader** system object function; this saves the video file to workspace to be used later. After which the **vision.OpticalFlow** was used to initialize a system object with various parameters, the computation algorithm used is the Horn-Schunck method[9]. The input data is then converted to single from unsigned eight bit integer as the output value is the form of complex vertical and horizontal components. Tracking is done with lines surrounding the objects of interest; this is because the output values used to track the vehicles are in complex form. Finally the algorithm is applied to every frame of the video sequence and the output video is analyzed. A few of the output frames for the tracked vehicles are shown in Fig.5;

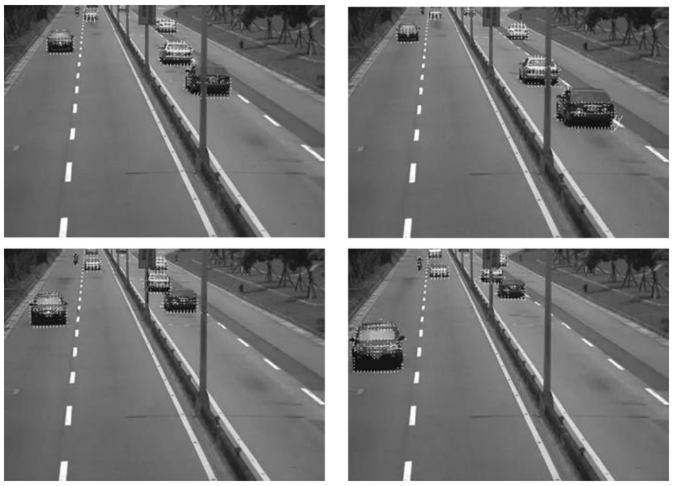


Figure 5. Video frames for object tracking using optical flow.

Observing the results in Fig.5 it can be concluded that the algorithm works flawlessly when it comes to tracking. The small white lines surrounding the vehicles in each of the frames are the bounding lines that are tracking the vehicles. However, when using the same algorithm for a video scene that has a bit of movement in its background the optical flow technique fails. There are also problems faced when implementing a counter in the algorithm as the output of the system object is in complex form, and it is difficult to relate and count all the lines tracking a particular vehicle. Even if the output is changed to magnitude squared, and the resulting output vectors for each vehicle is added or sized down, the vector value of each vehicle tracked is the same. This is because the number of lines specified to track any vehicle for all the successive frames is maintained a constant. If the objects of interest can be tracked using a bounding box a counter can be implemented, but this is not possible because the system object does not have bounding box coordinates as one of its output parameters. To resolve this problem a combination of two algorithms was used.

3.2 Algorithm2 - Optical flow and blob analysis

Introducing the general notion of Optical Flow as discussed in Section 3.1 in combination with Blob Analysis resulted in an effective mechanism to track and count the vehicles in the video sequence selected. To begin with, the video is read in to the workspace, then a system object for optical flow is initialized, after which the objects for the mean were declared to evaluate the vectors for the optical flow. A median filter was used to filter the noise, additionally two more morphological filters were also introduced one for filling the holes in the blobs, and the other to remove the stationary objects not wanted on the road. All the declared system objects were then used inside a loop to track the vehicles. The loop works by first converting each frame to grayscale and computing the optical flow, then a velocity threshold is calculated which is used to filter the video frame by using a median filter, the velocity vectors are evaluated by getting their magnitude squared as they are in complex form. A filter for filling up holes in the blobs is applied to all the frames. Values of area and bounding box of the blobs is then obtained and used to filter out objects that might not be vehicles. A counter is then implemented to count each vehicle. This is done by converting the column vector of the tracked vehicles into a thirty two bit integer that can be used to increment the counter every time a vehicle is detected. The counter is then displayed on the final tracking video sequence. Images for one of the video frames are shown in Figs.6-7; the images include frames with and without the filter, and when the vehicles are being tracked as shown in Fig. 8 and counted.

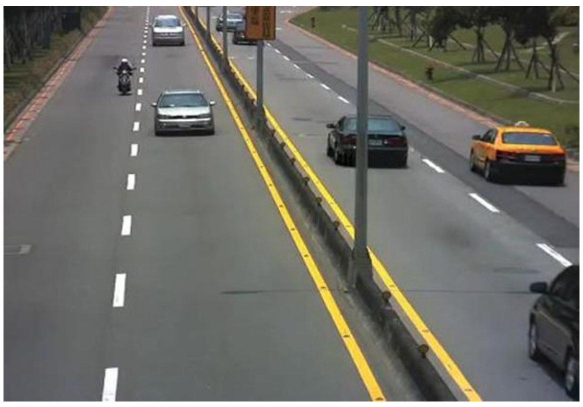


Figure 6. Algorithm2: Initial frame from the video.

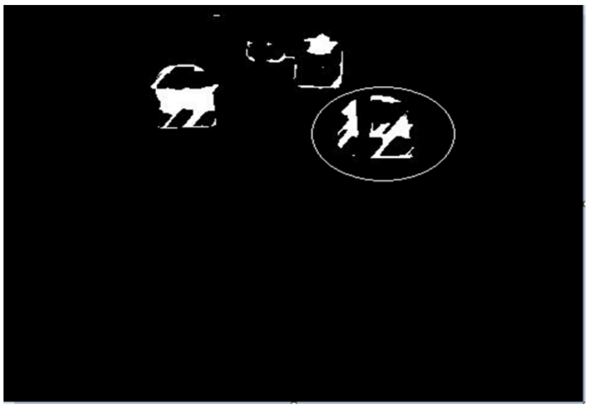


Figure 7. Algorithm2: Frame with filter applied

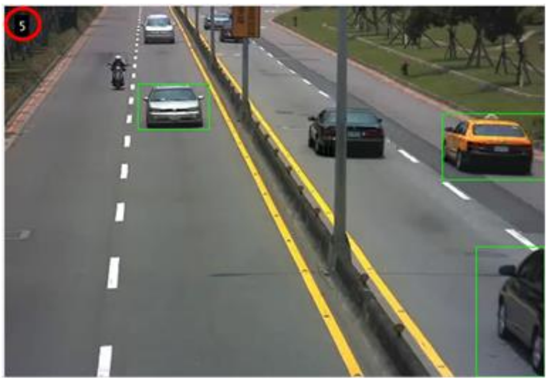


Figure 8. Algorithm2-OFBA: Frame with tracking and counting of vehicles.

From the results in Figs.6-8, it is seen how filtering aids in subtracting the noise and helps in tracking the vehicle more efficiently. The counter can be seen in Fig.8 on the top left side circled in red. There were also some difficulties faced with the algorithm despite the tracking being effective. This can be seen in Fig. 7, the white circle shows how the algorithm fails in tracking the vehicles when it passes through a light pole. A black bar kind of shape can be seen through the white blob of the circled vehicle. This means that the street light poles in the video are not taken as part of the background or as part of the road and subtracted in each frame.

5. TESTING AND RESULTS

Testing was divided in to two parts: first the counter was tested to check if it counted the vehicles tracked correctly. Secondly, the tracking of the vehicles was tested using a sequence of different videos.

4.1 Testing the Counter - Validation

The counter was tested on a MATLAB traffic video, because the vehicles were being tracked efficiently. This efficiency enabled the counter to increment every time a vehicle is tracked. At first when the code was run the vehicles shown on the counter were twenty seven which was in incorrect as the number of vehicles in the video were ten. To detect the problem the code for the counter was checked. A variable **totalvehicles** in the code adds the column vectors of the vehicles tracked and a 32 bit integer value is produced. This works in the sense that if two vehicles are tracked there will be two column vectors produced i.e. [1 1] and the variable adds this vectors to give two as the output. It was found out that there were some problems with the initial code, as it was updating this variable even if there were no vehicles tracked. The problem can be easily understood by looking at Figs.9-10;

totalcars	increment with current counter	totalcars	increment with current counter	totalcars	increment with current counter	totalcars	increment with current counter	totalcars	increment with current counter
0		1		1	9	2	18	2	
0		1		1		2		1	
0		0		1		2		1	26
0		1	2	0		1		1	
0		1		1		1	19	1	
0		2		1	10	2		1	
0		2	4	0		2	21	1	
0		2		0		1		0	
0		2		0		1	22	0	
0		2		0		1		1	
0		2		0		0		1	27
0		2		0		1		1	
0		2		0		1	23	1	
1	1	2		1		1		1	
1		2		1	11	1		1	
1		1		1		1		1	
1		0	5	1		2		1	
1		2	7	1		2	25	1	
1		1		1		2		1	
1		1	8	2	13	2		1	
1		1		3		2		1	
1		0		3	16	2		1	
1		0		3		2		1	
1		1		2		2		0	
1									

Counter Increment $\longrightarrow$

Figure 9. Increment of initial counter.

totalcars	increment with revised counter	totalcars	increment with revised counter	totalcars	increment with revised counter	totalcars	increment with revised counter	totalcars	increment with revised counter
0		1		1		2		2	
0		1		1		2		2	
0		1		1		2		1	
0		1		1		2		1	10
0		1		1		1		1	
0		1		1		1	7	1	
0		2		1		1		1	
0		2	3	0		1		1	
0		2		0		1		1	
0		2		0		1		1	
0		2		0		1		1	
0		2		0		1		1	
0		2		0		1		1	
1	1	2		0		1		1	
1		2		1		1		1	
1		2		1	5	1		1	
1		2		1		1		1	
1		2		1		2		1	
1		1		1		2	9	1	
1		1	4	1		2		1	
1		1		1		2		1	
1		1		2		2		1	
1		1		2	7	2		1	
1		1		2		2		0	
1									

Counter Increment $\longrightarrow$

Figure 10. Revised increment of the counter.

Fig. 9 shows how the initial counter increments from the left to the right of the figure when a vehicle is tracked. From the figure the counter is incrementing using totalvehicles, every time the totalvehicles variable is not equal to itself, an output of the same 32 bit integer is produced continuously as long as one or more vehicles are being tracked. This is seen on the figure where the output of the **totalvehicles** is constant for some time and then changes. An increment of the counter is shown beside the variable in the figure, and it can be observed that sometimes there are cases where the variable suddenly changes, and this change is not continuous for a certain amount of time as long as the vehicle is tracked. This change in effect increments the counter to the current twenty seven vehicles.

A revised table is made and the **totalvehicles** variable is changed where it is thought to have a wrong output. Fig.10 displays what the increment and output is thought to be, and at the end showing the correct number of total vehicles. As the counter was not functioning as expected because of unexpected outputs from the code, another code for the counter was written that was giving a close approximation of the total number of vehicles detected. This is how the improved counter works;

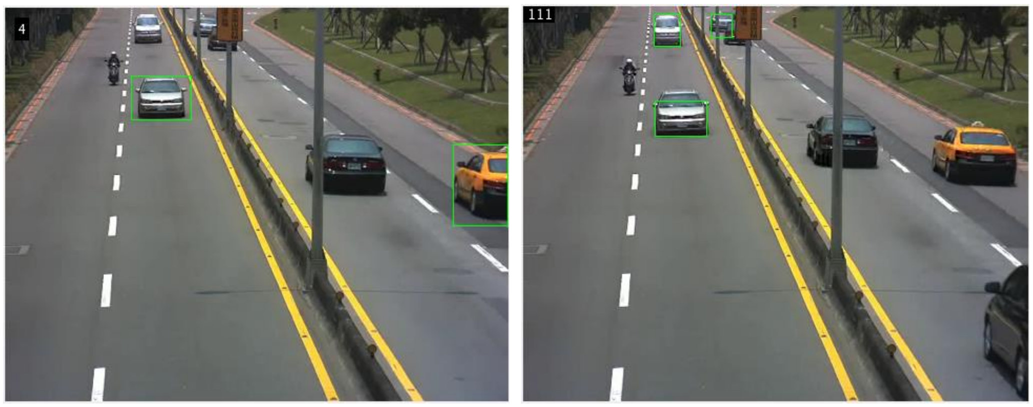
First two variables are initialized at zero; count and **totalvehicles**. Then it's checked if the output for the vehicles tracked is greater than **totalvehicles**, if true count is equal to vehicles tracked minus the total vehicles plus count, otherwise **totalvehicles** is equal to the vehicles tracked. This in sense updates the count only when output for the vehicles tracked is greater than total vehicles.

4.2 Testing the algorithm

A number of video sequences were tested with the two algorithms i.e. a combination of blob analysis and foreground detection, and a combination of optical flow and blob analysis. Both the algorithms were compared besides each other for their robustness in tracking the vehicles and counting them. There was no standard procedure set up to test the algorithm, so the tracking was visualized by eye and the counter was checked by manually counting the vehicles in the video sequence.

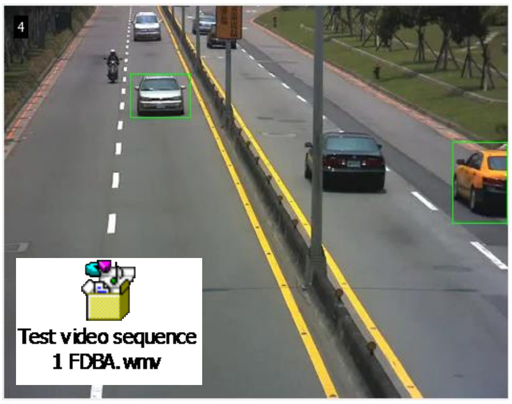
Test video sequence 1

This is the video sequence originally used when developing the code. The algorithm for the optical flow tracks the vehicles properly only when they are not passing through the street lamps. The foreground detection and blob analysis algorithm quite effectively tracks the vehicles and also counts the vehicles to an approximation closer to the total vehicles. A frame from both the algorithms is shown in Fig.11 for comparison.



(a) Algorithm1: Foreground detection and blob analysis (b) Algorithm2: Optical Flow and Blob analysis
Figure 11. Test 1, results of tracking for the two algorithms FDBA and OFBA respectively.

A video of the test video sequence 1 for Algorithm1 FDBA is shown below.



Video 1. Test video sequence 1 FDBA.wmv: http://dx.doi.org/doi.number.goes.here

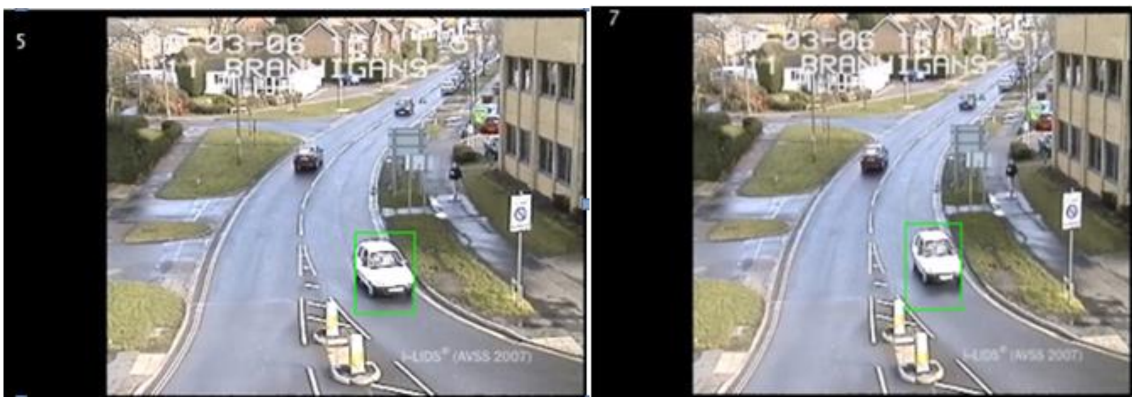
Fig.11 shows that in (b) all the vehicles are not tracked for the video sequence and the counter does not look right as it shows a count of 111 instead of 7 that is the correct number of vehicles in the frame. From analyzing the video it is seen that, when tracking the bounding boxes split up when a vehicle is tracked, or bounding boxes of two vehicles being tracked join together.

There are also problems observed with the counter, as it counts a vehicle multiple times increasing the final count. Looking at (a) it is seen that the vehicles are tracked properly when they are close than when they are far. Although the algorithm works well, there are some difficulties faced when two vehicles are close together, as the bounding box join together. On the other hand the counter is quite efficient as it give a close approximate of the original number of vehicles in the sequence.

<u>Test video sequence 2</u>

The sequence used for this second test is a clip from a surveillance camera on top of a lamp post. An advantage of this clip compared to test video sequence1 is that it does not contain any obstructions in the middle of the road like lamp posts. The first algorithm tracks the vehicles properly but does not count them correctly. This is because as there are shakes in the video scene because of the camera, and there are also instances in the scene when the bounding box breaks up and the counter starts counting the broken boxes. The images of the one of the frame are shown in Fig.12 for both the both the algorithms.

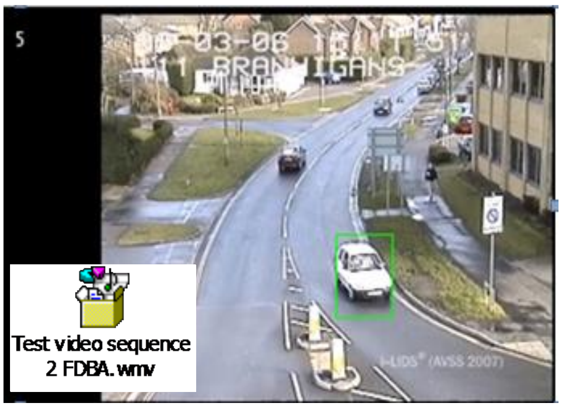
Examining Fig.12 (a) and (b), both the frames look similar for both the algorithms, but if we look closely the counter shows different total number of vehicles tracked. As discussed above the difference in the total count is due to the movement of the camera that breaks up the bounding boxes hence increasing the count. One of the other problems detected with the Algorithm2 - OFBA is that, the tracker has difficulties in tracking the vehicle when it turns towards the bend either to the left or coming in from the right. This is not the case with Algorithm1- FDBA. Otherwise both the algorithms are quite robust in tracking the vehicles.



(a) Algorithm1: Foreground detection and blob analysis (b) Algorithm2: Optical Flow and Blob analysis

Figure 12. Test 2, frames from both algorithms FDBA and OFBA after tracking.

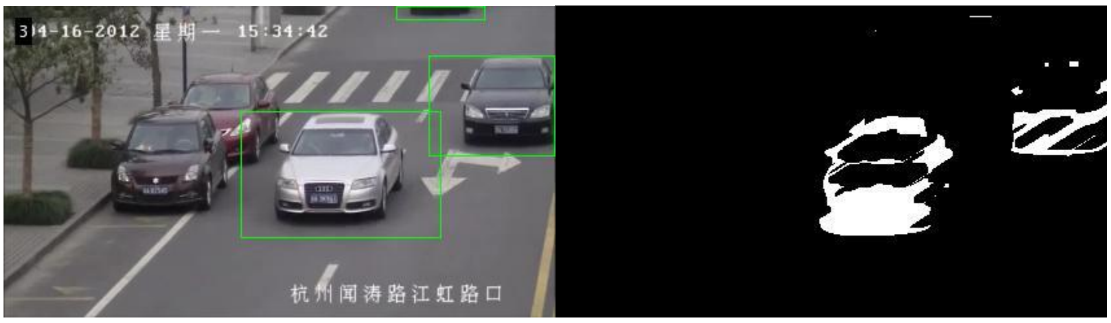
A video of the test video sequence 2 for Algorithm1 FDBA is shown below.



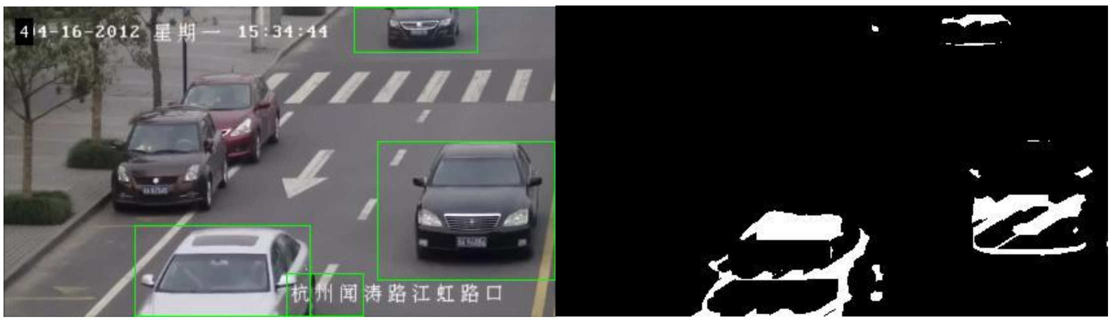
Video 2. Test video sequence 2 FDBA.wmv: http://dx.doi.org/doi.number.goes.here

Test video sequence 3

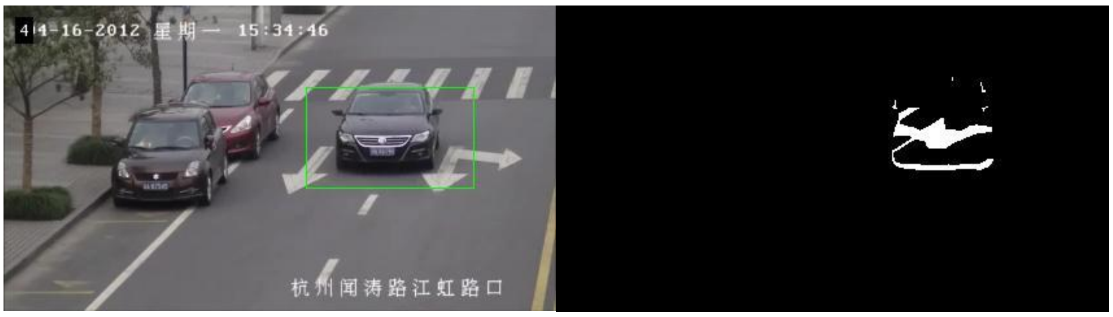
The third video for testing has got vehicles that are closer to the camera, this sequence tries to investigate if the algorithms work if objects are at close range. However one advantage of this video sequence is that the vehicles are only moving towards one direction that is towards the camera making it easier for the algorithms to cope with the computations better. The sequence is going to prove if the algorithms can handle moving objects up close, where a large area of the video frame has movement in it. Fig.13 shows the images of the tracking and thresholding results of the two algorithms proposed.



(a) (b)



(c) (d)

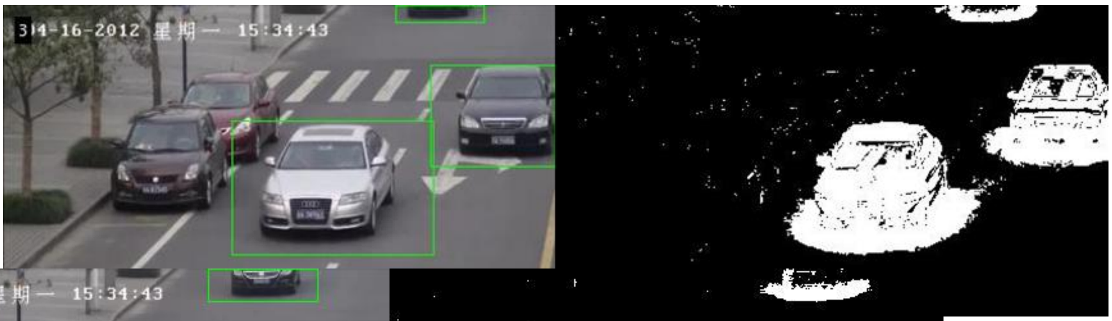


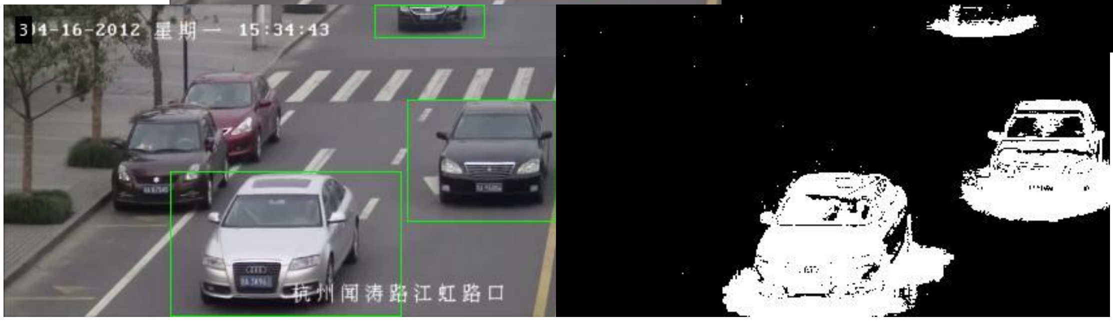
(e) (f)

Figure 13. Algorithm2 OFBA: Results of tracking and thresholding

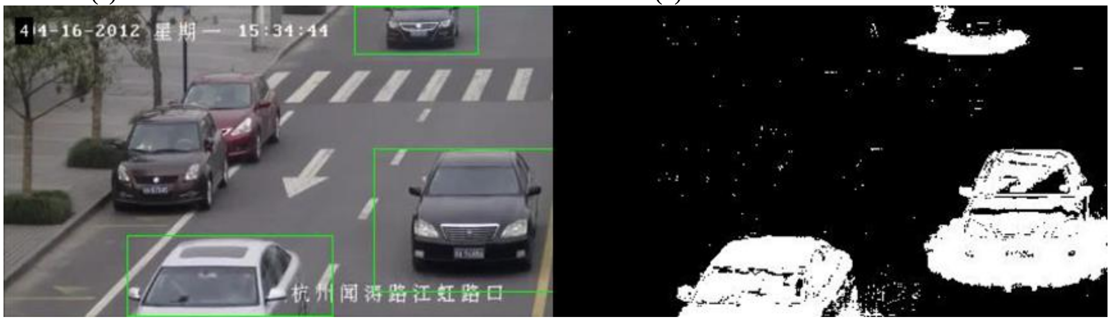
Fig.13 (a) shows the tracking of the vehicles, as seen there seems to be problems with it as the second vehicle at the back is not tracked at all in this frame compared to the first vehicle. When looked at the same frame for the thresholded image (b) it can be observed that the code is having difficultie

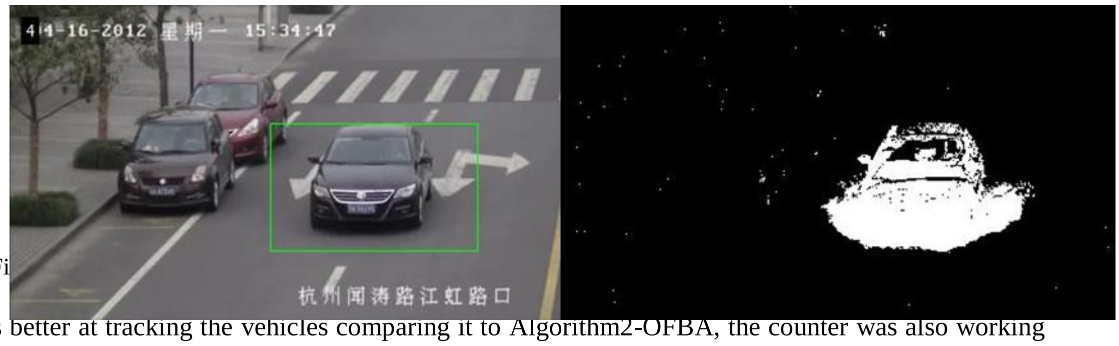
s to threshold vehicles that are dark colored, but it is easier for it to threshold light colored vehicles. This appears to be the case for the whole of the video sequence. When the vehicles get nearer the code finds it easier to threshold it, this means it tracks the vehicles, however it does not track it correctly as the bounding boxes are sometimes seen to appear and disappear in the entire video sequence. A reason for this might be that it is sometimes unable to differentiate between the road and the dark colored vehicles; and this would probably be because of the changing light conditions at the top of the frame and at the bottom. For this reason the end count is higher than the total number of vehicles. This can be solved by reducing the filter threshold so the vehicles will be detected more easily, and hence the tracking will be improved which in return improves the counting.





(c) (d)

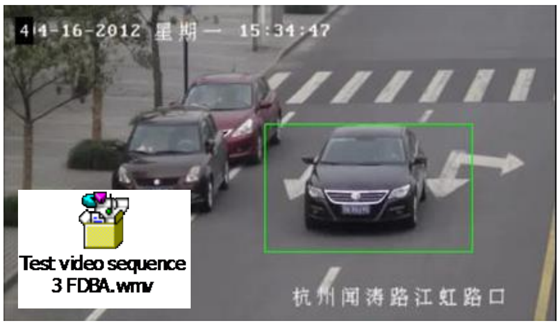




Fi

Algorithm1-FDBA was better at tracking the vehicles comparing it to Algorithm2-OFBA, the counter was also working as intended, giving a correct final value of the vehicles counted. Some of the images for the frames of the tracking and filtering for Algorithm2 are shown in Fig.15; The images show how the tracking takes when filtering is applied to the frames. It can be noticed that the algorithm is quite effective when it comes to tracking. Looking at Fig.14 (a) and (b) it is observed that when the vehicles are close by to the camera, the bounding boxes of nearby vehicles do not add up to become one, instead they keep on following the vehicle as long as it's in the frame. The filtering is also efficient as the whole vehicle is filtered out as a whole blob rather that some parts of it. This algorithm is noted to track dark colored vehicles equally as effectively as light colored vehicles.

There is however some problems noted when it comes to the total number of vehicles detected as it is not entirely accurate, but it is a better approximation in contrast to Algorithm2-OFBA. A video of the test video sequence 3 for Algorithm1 FDBA is shown in Video3.



Video 3. Test video sequence 3 FDBA.wmv: http://dx.doi.org/doi.number.goes.here

Test video sequence 4

A video sequence from a low resolution CCTV home surveillance camera is used to test the algorithm in this. This is done to check if the code can be used for low resolution camera for surveillance purposes. One major problem noticed with the video sequence is that the vehicles move faster as the video has a lower frame rate compared to other videos taken from high resolution cameras. Both the codes are tested using this video and the results are noted. Fig.15 shows how effectively the vehicles are being tracked; A video of the test video sequence 4 for Algorithm1 FDBA is shown in Video4.

When the results from the output video are analyzed, it is observed tracking for both the algorithms is similar as seen from the images, except that there are some frames where the second algorithm looses track of the vehicles. This might be because of the camera from which the video was obtained, overall the performance of both the codes is impressive. In spite of this the first algorithm still faces problems with the counter, the approximate of the number of vehicles is relatively high compared to that of the second algorithm.

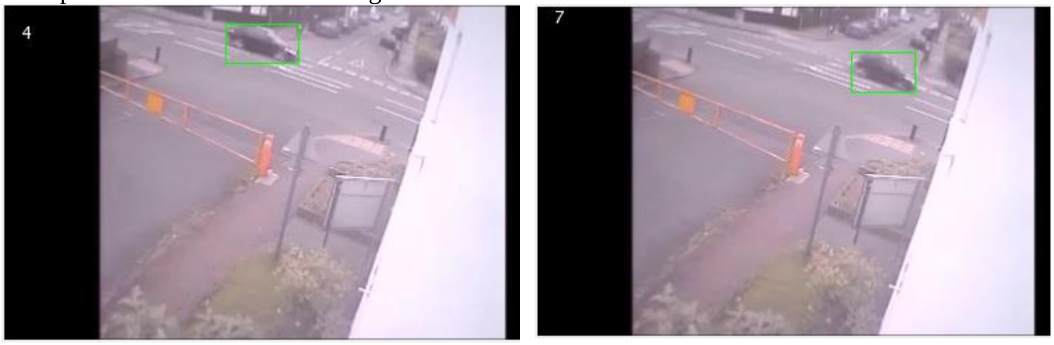


Figure 15: (a) Algorithm1 FDBA: Tracking results (b) Algorithm2 OFBA: Tracking results

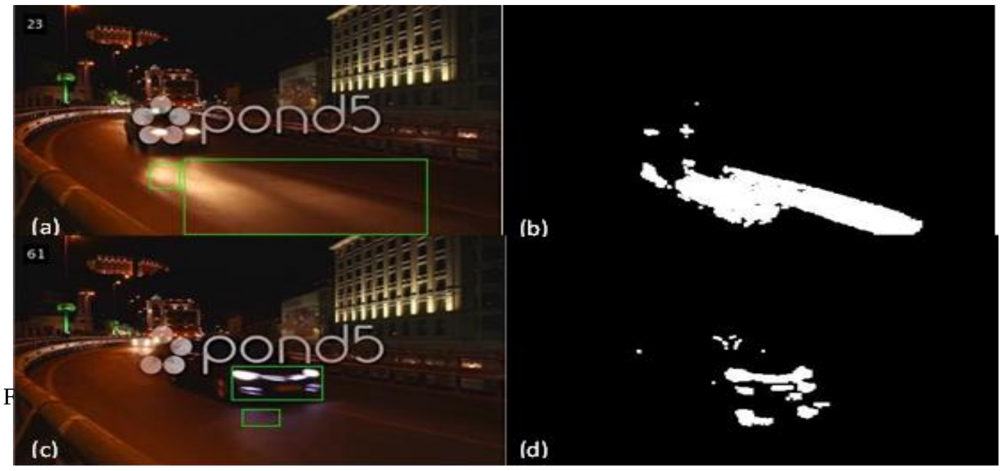
Video 4. Test video sequence 4 FDBA.wmv: http://dx.doi.org/doi.number.goes.here

Test video sequence 5

The video clip used in this test is when vehicles are passing through a junction bridge at night. This will be used to test if the algorithms are better or worse at performance when there is no day light or poorer light condition. Observing the images of the frame gained after the tracking will conclude this, and what are the reasons of their better of poorer performance. Images of the frames after tracking are shown below for both the algorithms.

Tracking results for Algorithm 1-FDBA is shown in Fig.16. Looking at (a) it is seen how the light from the headlights of the vehicle are tracked on the road below. The thresholding also shows when the foreground detection algorithm stops working as it cannot subtract the bright lights because it is in the foreground and not in the background. This might not be the case with tracking other objects that do not produce light when in the dark, but the algorithm does not work in this occasion.

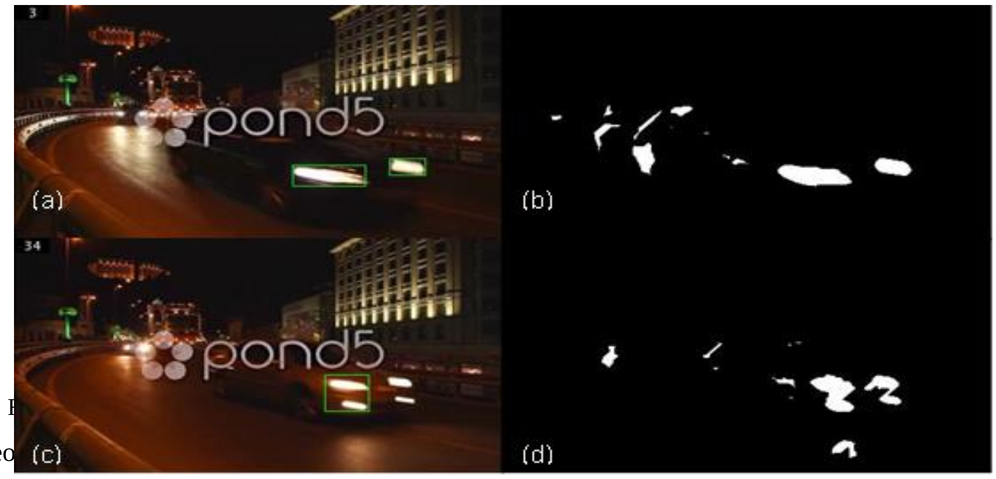
Tracking for Algorithm2 - OFBA does not look effective from the images shown in Fig.17. Image in Fig.17 (a) shows the tracking frame and (b) shows the thresholding for the same frame, this is similar for (c) and (d). It's seen that the algorithm is not capable of tracking the vehicles in low light conditions or during the night. Observing (a) and (b) it is realized that only the headlights of the vehicles are being tracked, even looking at the thresholded image it can be deduced that the filters are only filtering the objects that are brighter. The optical flow and blob analysis algorithm uses the Horn–Schunck method to compute optical flow; this method calculates the optical flow by using the apparent velocity of the brightness patterns in an image[9]. It can be concluded that both the algorithms are unable to cope with tracking vehicles in the dark.

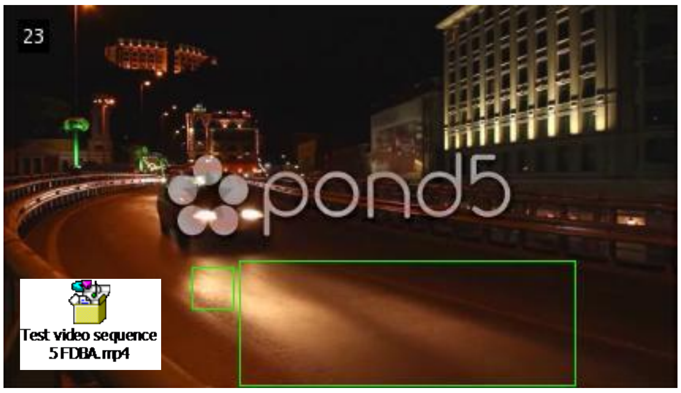


F

F

A video of the test video





Video 5. Test video sequence 5 FDBA.wmv: http://dx.doi.org/doi.number.goes.here

6. CONCLUSION AND FURTHER WORK

The main aim of this work was to develop an algorithm that can track vehicles, which has been successfully prototyped using various methods and a selection of such algorithms were implemented for further analysis. Two algorithms were further analyzed namely Blob Analysis and the combination of Blob Analysis with Optical Flow.

The work revealed that vehicle tracking is a non-trivial problem. The combined algorithm did improve tracking performance; however other minor problems were seen to be manifested that requires to be addressed such as camera motion.

Testing of the algorithms was done for a number of different video sequences that each contains a different challenge from the next. The results were then analyzed to identify why the algorithms failed to track the vehicles. Reviewing the test results the conditions when the algorithms are likely to be inaccurate or inefficient can be summed up as follows;

- Both algorithms have difficulties with occlusions, this is noted in the first test where the poles come on the way when detecting and tracking the vehicles.
- Problems in tracking due to movement of the camera, as seen in test two.
- Complications with thresholding, as observed with the second algorithm. Dark colored vehicles are not thresholded suitably compared to the light colored vehicles, this led to inefficiency in the tracking.
- For the fourth test, FDBA encountered problems in tracking as it loses track of some vehicles due to low resolution of the camera from which the video sequence was obtained. It can be concluded that the algorithm might not be suitable for tracking objects in a low resolution video.
- Both the algorithms in the last test fail to track the vehicles in low lighting.

The algorithms need to be tested on a larger video database to examine their performance when it comes to background motion and changing light conditions. Improvements still needs to be done to both the algorithms so they are both able to perform in the presence of object occlusions and/or a moving camera. With these enhancements the algorithms could be used further for tracking generic vehicles instead of only vehicles.

7. REFERENCES

[1] Guiseppe Salvi, "An Automated Vehicle Counting System Based On Blob Analysis For Traffic Surveillance", Proc., International Conference On Image Processing, Computer Vision And Pattern Recognition, 397-403, ISBN 9781601322258, (2012).

[2] Song, X. And R. Nevatia, "Robust Vehicle Blob Tracking With Split/Merge Handling", In Proceedings Of The 1st International Evaluation Conference On Classification Of Events, Activities And Relationships, Springer-Verlag: Southampton, UK. P. 216-222, (2007).

[3] V Kastrinaki, M Zervakis, K Kalaitzakis, "A survey of video processing techniques for traffic applications", Image and Vision Computing, Volume 21, Issue 4, 1 April 2003, Pages 359-381, ISSN 0262-8856, http://dx.doi.org/10.1016/S0262-8856(03)00004-0,(2003).

[4] Thou-Ho Chen; Yu-Feng Lin; Tsong-Yi Chen, "Intelligent Vehicle Counting Method Based on Blob Analysis in Traffic Surveillance," *Innovative Computing, Information and Control, 2007. ICICIC '07. Second International Conference on* , vol., no., pp.238,238, 5-7 Sept. (2007).

[5] Goo Jun, J.k.Aggarwal and Muhittin Gokmen, "Tracking and Segmentation of Highway Vehicles in Cluttered and Crowded Scenes", IEEE 2008 Workshops on Applications of Computer Vision, Copper, Colarado, WACV'08, 1-6, (2008).

[6] Fatemah Karimi Nejadasi, Ben G.H. Gorte and Serge P. Hoogendoorn, "Optical flow based vehicle tracking strengthened by statistical decision", ISPRS Journal of Photogrammetry and remote Sensing, Vol. 61, Issue 3-4, 159-169, Dec. (2009).

[7] Gern, A.; Moebus, R.; Franke, U., "Vision-based lane recognition under adverse weather conditions using optical flow," *Intelligent Vehicle Symposium, 2002. IEEE* , vol.2, no., pp.652,657 vol.2, 17-21 June 2002 doi: 10.1109/IVS.2002.1188025.

[8] The MathWorks, I. Computer Vision System Toolbox. 1994-2014, 22 Mar.2014, http://www.mathworks.co.uk/help/vision/index.html.

[9] Horn, B.K.P. and B.G. Schunck, "Determining Optical Flow", Artificial Intelligence (17), 185-203,(1981).