Traffic Flow Estimation

Muhammad Asad
Reg No. 110128953
Department of Electrical and Electronics Engineering
University of Sheffield
UK

I. Objective

The objective of this project is to design and implement a real-time system which can estimate traffic flow using different computer vision techniques. The implementation is done using OpenCV libraries on beagleboard, making the system both real-time and portable.

II. Introduction and Background study

With the recent advancements in the field of Computer Vision and technology, it has become possible to address different complex real-world problems using portable devices. One such extensively researched problem is to monitor traffic in real-time.

Previously magnetic loops were used to calculate different parameters of traffic. These systems were later replaced by Vision based systems, which were able to calculate more parameters (such as mean speed, density and flow) which can be processed to achieve more accurate and detailed results [1]. Moreover these sensors were easy to install as they were just mounted on tall poles, as compared to cumbersome installation process of magnetic loops.

The introduction of computer vision based techniques posed some new challenging problems. One such problem is to estimate the traffic flow in real-time. Traffic flow estimation involves the use of different traffic parameters such as speed and number of cars to estimate the flow. This estimated flow can then be used in traffic planning to stop congestion by diverting traffic to empty roads.

In literature a lot of similar work has been done to estimate different parameters of the traffic. Based on the approach used, these proposed methods can be listed into three different categories.

1) **Background Modelling based methods:** These type of methods involve modelling of the background using different techniques. The modelled background is then used to extract the foreground object by subtracting the background frame from the original frame. These foreground objects are then used for calculation of different parameters related to traffic including traffic flow [2] [3] [4]. The main advantage of using these approaches is that they are adaptive to the changes in the scene, which is automatically updated into either foreground or background depending on its motion. On the contrary a common disadvantage of using these methods is that due to limited 2D view of the cameras, it becomes difficult to differentiate between two objects if they are close to each other. It also relies on motion, therefore if there is no motion, this technique fails.
2) **Feature based methods:** These methods use the distinctive features from a feature extraction technique, to track and model the traffic on roads. These approaches suffer from a lot of different problems. First it is difficult to differentiate between features of cars that are of the same model and same colour. Secondly, even if they are identified correctly, small inter-frame changes cause a lot of problem. For these methods, physical appearance of cars greatly affects the output [5] [1].
3) **Optical Flow based:** The approaches that lie under this category use optical flow based motion estimation to analyse the traffic and calculate different related parameters. These approaches are highly accurate in estimating the parameters [6], however these type of approaches use a lot of computational power. A major requirement to build a real-time system is to use computationally less expensive techniques, therefore these methods are not widely used in real-time systems.

A lot of approaches have been proposed to cater for each of the problems listed above. In [7], an approach is proposed which is invariant to problems arising from camera view angle. A similar approach was also proposed in [8] and [9], where the author uses rectified images to estimate the traffic parameters successfully. These rectified images are projection of the original images, used to get the bird-eye view of the traffic. All of these approaches have a common limitation. They all required at least one actual distance parameter from the road, i.e. width or length of the road, width of the lanes, camera angle etc. These parameters are not always present, and in case there is slight change in camera orientation these methods will fail.

Although there have been a lot of attempts previously to estimate different parameters of traffic, there has not being many attempts to estimate the traffic flow in real-time using different complex scenarios. Almost all of the approaches previously proposed did not use any portable device, like beagleboard, to make a real-time system. Most of

the approaches used full system computational power, with controlled simple scenarios.

This project proposes a real-time approach which can be used on portable devices like beagleboard to successfully estimate the flow of traffic. This method is presented and discussed in subsequent sections. Section III depicts the proposed approach, while Section IV presents the results. In Section V an analysis on the performance and working of the proposed approach is given. This report concludes with conclusion and future work in Section VI.

III. Method

The proposed approach uses both speed and number of cars to estimate the traffic flow in real-time. To achieve this a model based approach is used. Two models are used for two different tasks in this project. To estimate the background, a background modelling technique is devised, whereas to estimate the traffic flow, a traffic model is constructed to map the traffic into different parameters in the program. These models are connected using different processing steps in between, which helps to address each of the problems faced while processing from inputs. These problems include, distorted output from segmentation of vehicles in the foreground, disconnected foreground components for the same vehicle and tracking and estimating speed of each car. The input frames are first cropped to a size of 320 by 100 pixels at the region where the cars are closer to the camera and can be easily modelled. This also helps in making the system real-time as with smaller frame, less computation is required. This cropped frame is then fed to a system whose flowchart is shown in Figure 1. Each of the steps involved in this system

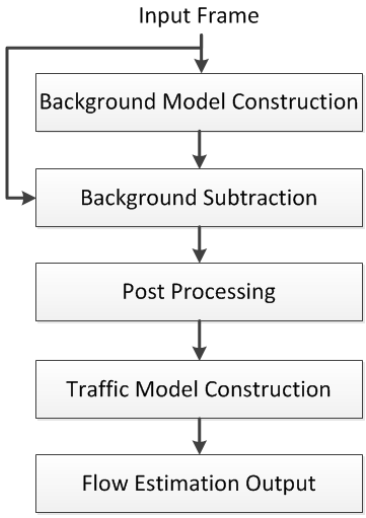


Fig. 1: Flow Chart of the system

are discussed below in subsequent sections.

A. Background Model Construction

Background modelling is used to construct a background, which is then subtracted from original image to get the foreground mask. Recursive techniques update a single background image based on input frame. These techniques include Kalman filter, Mixture of Gaussian (MOG) and median filter. They are computationally less expensive, however they are not very robust to calculate the background image as compared to other Non-recursive techniques used. Non-recursive technique involve estimating a background model by using statistical properties of different frames over time. These include techniques like Frame differencing, Median Filter, Non-parametric and Linear predictive filters [10]. Most of these background modelling techniques are computationally complex and use a lot of memory. However these technique area accurate and robust, as they use very less number of frames to estimate the background. To implement a similar technique on beagleboard would have made it impossible to estimate the traffic flow in real-time. Therefore we devised a background modelling technique which uses some basic ideas from different previously developed techniques. The proposed background modelling technique is much more efficient and robust, with very low computational complexity. This step of the system is the most important step in making it a real-time system, as a good background model results in a good estimation of the flow.

The background model is constructed using a collection of linked lists which are equal to the number of pixels in the frame region. Each linked list represents a set of significant pixel values that have occurred over time at a specific location. Each node in this linked list contains an RGB pixel value along with its significance. This significance value is directly responsible for robust and accurate background estimation.

The algorithm used for construction of this model works on three conditions. For first image, heads of all the linked lists are initialized with corresponding RGB value and a starting value for significance of 10. For next images, the RGB value at each location is given as an input for a corresponding linked list. This value is compared to the existing nodes in that list. If there is an exact match found for all of the three channels, then the significance of that matched node is increased by a value of 10. If all the RGB values lie within a range defined by a threshold t then the significance value of the corresponding node is increased by 5. The last condition is if there is no match, for this condition significance of each unmatched node is subtracted by a penalty of 2 and a new node with input RGB value is added to the linked list with an initial significance value of 10. Maximum significance, which a given pixel node can have, is set to 200. This value actually represents the sensitivity, of the modelling algorithm, to the changes in the background. Setting this value too low can result in an inaccurate background. On the contrary, if this value is set too high then the model is not so adaptive to changes in background. It, then, takes a lot of frames to estimate the change. The value of 200, which is currently assigned, is found by using different values and observing the output. This value is the most suitable as it is neither too high nor too low, resulting in both an adaptive and accurate background estimation. This algorithm is depicted below.

Algorithm 1 BACKGROUND_MODELLING(R, G, B)

1: Iterate through nodes with value(R_i, G_i, B_i)
2: **if** $(R_i == R, G_i == G, B_i == B)$ **then**
3: Update $Sig_i = Sig_i + 10$
4: **else if** $(R_i - t < R < R_i + t, G_i - t < G < G_i + t, B_i - t < B < B_i + t)$ **then**
5: Update $Sig_i = Sig_i + 5$
6: **else**
7: Update $Sig_i = Sig_i - 2$
8: **end if**

The value of threshold t can be adjusted to increase the accuracy, however there is a trade off of robustness to accuracy. If t is kept too low, it gives a very accurate background model, however it takes more frames to complete the background estimation. On the contrary, if t is kept high, the estimated background is not so accurate, however it takes less number of frames for the estimation process.

The background model also keeps track of unused RGB nodes. Due to the modelling algorithm, the significance of such unused nodes drop below the value of zero, and at each update if a node has a value less than zero then it is deleted from the list. This saves a lot of memory, and keeps the lists short and manageable. In this way, only the most significant value, i.e. the values that represent the background and have high probability of occurring again are kept in the model along with their significance value. This process of background estimation is generates and updates background for every new frame in the sequence. Therefore it is robust in keeping the background as close as possible to the background in actual sequence.

B. Background Subtraction

This step works using a simple but effective method. It is noticed that after extraction of the background, subtracting the background image with the current frame result in a lot of negative values. As a result of thresholding, these values are not included in the final result. This makes the output binary image of the detected blobs to contain many distorted blobs, with a lot of discontinuities due to these negative detection values. To cater for this problem, an approach is proposed which performs both thresholding and subtraction in one step. This approach is similar to average frame differencing approach from [11], however the complexity of the method is further decreased.

The idea is to use two thresholds (high and low threshold) instead of using only one. The background image is received from the background model. Using this background image a comparison is done between background and current frame to detect the foreground. In each of the RGB channels, the values of the background image are compared to the current frame and an output foreground mask is generated. The values which are greater than the value of sum of background and high threshold and those values which are less than the value of difference of background and low threshold correspond to a foreground object detection. For the rest of the values, which lie within the range defined by thresholds and background correspond to the output background. This process is described in Figure 2 using a 1D image signal for background. Both high threshold and low threshold are shown in this figure. A value of +60 is used for high threshold while low threshold is assigned a value of -60. This value is used to make three regions for decision as depicted in Figure 2.

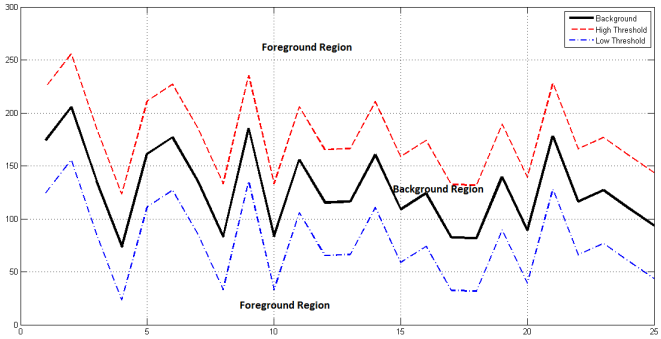


Fig. 2: Background Subtraction Using Thresholds

C. Post Processing

The output foreground mask from background subtraction in the previous step has a lot of disconnected foreground objects. Therefore they require a post processing algorithm, which can join them to form one blob. This step makes it easy to detect and model a car for the step which follow.

To correct this problem, morphological closing operation is used. This operation performs image dilation followed by erosion, using the same structuring element. Since most of the cars have a shape similar to a square, therefore a square shaped structuring element of 3x3 is used. This helped in improving the results a lot, as most of the discontinuities in the foreground mask are square or rectangular because of the shape of the cars and different objects on the cars like windows, mirror etc. Dilation connects the disconnected regions by filling the holes, while distorting the objects in all direction. To correct this distortion, an equivalent erosion function is performed.

D. Traffic Modelling

The output from post processing is used to construct a traffic model. This step of the system involve performing two operations using the foreground mask. First operation is to extract the bounding box and coordinates of the centroid of each car. Second operation is to use this data in temporal space to estimate number of cars per frame and average speed of cars on the road.

The extraction process uses contour scanner in OpenCV [11]. A perimeter constraint is added to make sure that only object equal to or larger than the size of a car get extracted. The contours are extracted and their edges are smoothed using a polygon approximation method in OpenCV. The bounding box for each contour is found by using OpenCV function

cvBoundingRect. To find the center(x_c, y_c), moments M00, M10 and M01 are used in the following equations [11].

$$x_c = \frac{M10}{M00}, y_c = \frac{M01}{M00} \quad (1)$$

These parameters are used to create a model of each car. This is done using only one linked list which represents the traffic, while each individual node on this list represents a car. The parameters stored for each car include coordinates of a car's centroid, height and width of the bounding box, a timer (which is used to update status), frame ID (for checking which frame the car is existing in), speed and status of the car (to check whether it is active or not).

The parameters for each centroid, extracted from each frame are fed to the traffic model algorithm. This algorithm first checks if the car already exists in the previous frame. This is done using the simple observation that if a car is travelling its centroid always remains inside the bounding box, for the same car, from previous frame. There is only a small deviation $(\Delta x_c, \Delta y_c)$ in the centroid. This step also uses a unique frame ID given to each frame to check which frame is the car from. If a car is successfully identified then its speed, centroid location, width and height of bounding box and status is updated. The calculation of speed is done using Euclidean distance between current and previous centroid location and using time calculated for two frames using FPS value(frames per second) from the video sequence. This algorithm is presented below:

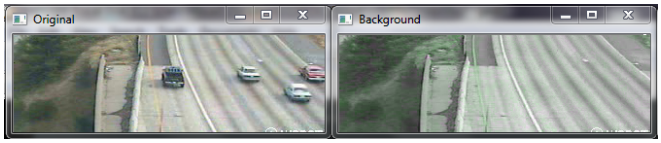
Algorithm 2 TRAFFIC_MODELLING
$(x_c, y_c, width, height, frameID)$

1: Iterate through nodes with value $(x_i, y_i, width_i, height_i, frameID_i)$
2: **if** $(x_i - width_i/2 < x_c < x_i + width_i/2, y_i - height_i/2 < y_c < y_i + height_i/2)$ **then**
3: Check $frameID$
4: **if** $(frameID \neq frameID_i)$ **then**
5: Update $Speed, Center(x_i, y_i)$, $width_i$, $height_i$ and $frameID_i$ using input
6: **end if**
7: **else**
8: Add new car node with input parameters
9: **end if**

E. Flow Estimation Output

The modelled traffic is then used to estimate the flow. This step uses the average speed of all the cars on the road along with the total number of cars per frame to estimate the flow of traffic.

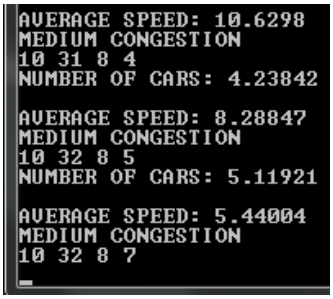
This step works by using a process similar to feature voting, however here voting is done according to the value of each of the two parameter, i.e. average speed and no. of cars. Four counters are initialized, one for each situation. These four counters collect the vote for each parameter falling in the range of each flow condition. The corresponding flow to the highest counter value is declared the flow of the road. This method uses a maximum value for these counters. When this value is reached by one of the counters, and it try to go above it, value of each of the other counters is decremented until they reach 0. This method is based on the observation that neither of the flow condition stays forever. The speed and number of cars on the road are constantly changing, however the flow is an overall estimation of the maximum occurring condition on the road. At an instant there will be more than one counters with value greater than zero, this shows exactly the same situation. However, the dominating counter for vote will be the one having highest value. Hence giving an estimation of an overall flow.



(a) Original Frame with its Modelled Background

(b) Background Subtraction and Post Processing

(c) Traffic Model Output



(d) Traffic Flow Output

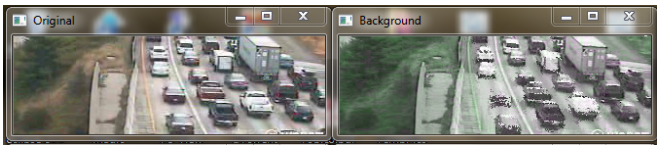
Fig. 3: Proposed System's Output for Sequence with Medium Congestion

IV. RESULTS

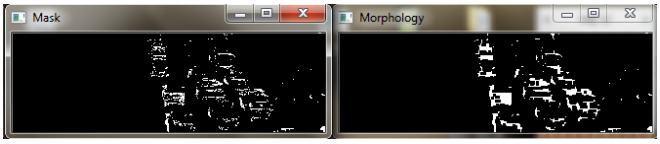
The proposed system successfully identified four different categories of traffic congestion with complete accuracy. It uses first few frames to estimate the background, over which it is unable to estimate the flow. However as soon as it gets background model, it is able to accurately estimate the traffic flow. This algorithm is tested using beagleboard and it successfully runs in real-time with an average of frame rate of $4.8fps$.

Figure 3 present the results of each of the steps in system for medium congestion sequence, whereas Figure 4 presents

the results using high congestion sequence. The result can be seen in the output window (Figure 3(d) and Figure 4(d)). It gives the average speed of the road, number of cars per frame and output flow estimation in the form of type of congestion. It can be observed that this system has dependency on motion, therefore if there is no motion in the sequence, then the flow estimation becomes difficult.

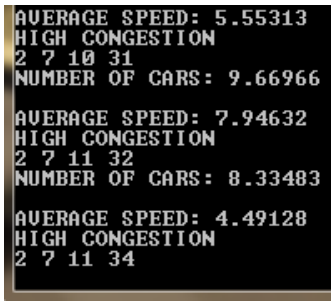


(a) Original Frame with its Modelled Background



(b) Background Subtraction and Post Processing

(c) Traffic Model Output



(d) Traffic Flow Output

Fig. 4: Proposed System's Output for Sequence with High Congestion

V. Discussion

The proposed approach for traffic flow estimation performs well in almost all situations, however it still has some limitations. These limitations will be discussed in this section, along with the analysis of overall system performance.

The algorithm uses the fact that if there are cars on road, then there is always some movement present. This assumption is used to build a model for the background. This background model uses everything that is stationary in the frame to model the background. Whereas every moving object is identified as foreground object. An advantage of using this approach is that it is highly adaptable to changes in background, which can occur in a real-world scenario. These changes can arise from changes in different physical constraints such as changes in weather, slight camera tilt or angle changes or even addition of different object to the scene such as sign boards or parked cars. These changes are successfully catered for by the background modelling technique, which is able to adjust the model within 5 frames depending on the max limit for significance value. A limitation associated with this advantage is that the system is unable to detect stationary cars in the scene. This condition might be true if there is very high congestion on the road, resulting in a lot of stationary cars. If this happens then all cars will be added to the background, resulting in nothing on the output. The system, however, works well even if there is slight movement. This is evident from the output in Figure 4, where even with only small movement detection, the algorithm is successful in identifying the congestion correctly. In real-world scenarios, there is always some movement which exists even in the highly congested traffic. However, there is still still a probability of all cars becoming stationary.

Another limitation of this system is that it needs few frames for initial model construction. Once constructed, this model is then updated in real-time as the program runs. The output is given in real-time, and if there is a transition between different congestions, then the system is able to detect this successfully and estimate the traffic flow with accuracy at an average of frame rate of $4.8fps$.

VI. Conclusion

A real-time system for estimating traffic flow has been proposed in this work. This system has been implemented on beagleboard, which made it portable as well. This system accurately estimated traffic flow in all of the given sequences. However, the proposed approach is dependent on motion, therefore it does not work in situations where there is no motion on the road. Apart from that, proposed approach is adaptive to the changes in scene and the implementation uses very less computational power. This system is able to produce an average frame rate of $4.8fps$ on beagleboard, which makes it a real-time system.

Future work involves improving the approach to cater for situations where there is no movement. A possible solution can be to use an edge detector. This system can further be improved with different machine learning techniques to make it more adaptive, so that it learns from its mistakes.

References

[1] D. Beymer, P. McLauchlan, B. Coifman, and J. Malik, "A real-time computer vision system for measuring traffic parameters," in *Computer Vision and Pattern Recognition, 1997. Proceedings., 1997 IEEE Computer Society Conference on*, pp. 495–501, IEEE, 1997.

[2] M. Vargas, J. Milla, S. Toral, and F. Barrero, "An enhanced background estimation algorithm for vehicle detection in urban traffic scenes," *Vehicular Technology, IEEE Transactions on*, vol. 59, no. 8, pp. 3694–3709, 2010.

[3] D. Gao and J. Zhou, "Adaptive background estimation for real-time traffic monitoring," in *Intelligent Transportation Systems, 2001. Proceedings. 2001 IEEE*, pp. 330–333, IEEE, 2001.

[4] H. Liu and W. Chen, "An effective background reconstruction method for complicated traffic crossroads," in *Systems, Man and Cybernetics, 2009. SMC 2009. IEEE International Conference on*, pp. 1376–1381, IEEE, 2009.

[5] X. He and N. Yung, "Corner detector based on global and local curvature properties," *Optical Engineering*, vol. 47, p. 057008, 2008.

[6] Z. Zhang, Y. Hou, Y. Wang, and J. Qin, "A traffic flow detection system combining optical flow and shadow removal," in *Intelligent Visual Surveillance (IVS), 2011 Third Chinese Conference on*, pp. 45–48, IEEE, 2011.

[7] L. Grammatikopoulos, G. Karras, and E. Petsa, "Automatic estimation of vehicle speed from uncalibrated video sequences," in *Proceedings of International Symposium on Modern Technologies, Education and Profeesional Practice in Geodesy and Related Fields*, pp. 332–338, Citeseer, 2005.

[8] C. Maduro, K. Batista, P. Peixoto, and J. Batista, "Estimation of vehicle velocity and traffic intensity using rectified images," in *Image Processing, 2008. ICIP 2008. 15th IEEE International Conference on*, pp. 777–780, IEEE, 2008.

[9] C. Maduro, K. Batista, and J. Batista, "Estimating vehicle velocity using image profiles on rectified images," *Pattern Recognition and Image Analysis*, pp. 64–71, 2009.

[10] S. Mohamed, N. Tahir, and R. Adnan, "Background modelling and background subtraction performance for object detection," in *Signal Processing and Its Applications (CSPA), 2010 6th International Colloquium on*, pp. 1–6, IEEE, 2010.

[11] G. Bradski and A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, 2008.