Object Detection and Tracking for Traffic Surveillance

Kavya K R,
Dept of Instrumentation Tech,
Dayananda Sagar College of Engineering.

Divya C Sejekan
Dept of Instrumentation Tech,
Dayananda Sagar College of Engineering.

Sneha Rani R
Dept of Instrumentation Tech,
Dayananda Sagar College of Engineering.

***Abstract*—Traffic management is becoming one of the most important issues in rapidly growing cities. Due to bad traffic management a lot of man-hours are being wasted. Increasing congestion on highways and problems associated with existing detectors has generated an interest in vehicle detection technologies such as video image processing. Automated motion detection and tracking is a challenging task in traffic surveillance. In this paper, a system is developed to gather useful information from stationary cameras for detecting moving objects in digital videos. The moving detection and tracking system is developed based on optical flow estimation together with application and combination of various relevant computer vision and image processing techniques to enhance the process. To remove noises, median filter is used and the unwanted objects are removed by applying thresholding algorithms in morphological operations. Also the object type restrictions are set using blob analysis. The results show that the proposed system successfully detects and tracks moving objects in urban videos.**

Keywords—Optical flow estimation, moving object detection, tracking, morphological operation, blob analysis, MATLAB simulnik.

I. INTRODUCTION

The objective of this project is to identify and track moving vehicles within a video sequence. The tracking of the vehicle is based on optical flows among video frames in contrast to image background-based detection. The proposed optical flow method is straightforward and easier to implement and we assert has better performance. The project consists of software simulation on Simulink and can be implemented as hardware on TI TMS320DM6437 DSP board. The idea of this project is derived from the tracking section of the demos listed in MATLAB computer vision toolbox website. The Simulink model for this project mainly consists of three parts, which are “Velocity Estimation”, “Velocity Threshold Calculation” and “Vehicle Boundary Box Determination”. For the velocity estimation, we use the optical flow block in the Simulink built in library. The optical flow block reads image intensity value and estimate the velocity of vehicle motion using either the Horn-Schunck or the Lucas-Kanade. The velocity estimation can be either between two images or between current frame and Nth frame back. We set N to be one in our model. After we obtain the velocity from the Optical Flow block, we need to calculate the velocity threshold in order to determine what is the minimum velocity magnitude corresponding to a moving object. To obtain this velocity threshold, we first pass the velocity through couple mean blocks and get the mean velocity value across frame and across time. After that, we do a comparison of the input velocity with mean velocity value. If the input velocity is greater than the mean value, it will be mapped to one and zero otherwise. The output of this comparison becomes a threshold intensity matrix, and we further pass this matrix to a median filter block and closing block to remove noise. After we segment the moving vehicle from the background of the image, we pass it to the blob analysis block in order to obtain the boundary box for the object and the corresponding box area. The blob analysis block in Simulink is very similar to the “region props” function in MATLAB. They both measure a set of properties for each connected object in an image file. The properties include area, centroid, bounding box, major and minor axis, orientation and so on. In this project, we utilize the area and bound box measurement. In our model, we only display boundary box that is greater than a certain size, and the size is determined according to the object to be track. The rest of the Simulink model should be self explanatory.

II. EASE OF USE

A. *DESIGN AND IMPLEMENTATION*

In this Simulink model, there are couples of major parameters that we need to adjust depending what the tracking object is. The first parameter is the gain after the mean blocks in the velocity threshold subsystem. If too much background noise besides the moving objects is included in the output intensity matrix, the gain need to be adjust to filter out background in the image. The second parameter is the constant that is used for comparison with the boundary box. Any boundary boxes with area below this constant is filter out. One of the disadvantages of optical flow based tracking is that a moving vehicle may have many small boundary boxes due to the optical detection on different part of the moving vehicle. In order to better keep track of the moving vehicle, we need to filter out the small boundary boxes and keep the large boundary box. The other minor parameters such as the shape for the display of motion vector and tracking box are up for the users to decide.

B. *METHODOLOGY*

The algorithm has following stages,
1) Feed a video file to be tracked as an input.
2) Convert color frames of video to grayscale video frames.
3) Compute optical flow between current frame and Nth frame back
4) From above step we can calculate velocity of motion vectors.

5) Out of all pixels of the frame only moving pixels are of moving object.
6) Compute magnitude of vector of velocity which can we get through optical flow & take a mean.
7) Use median filter to get threshold image of moving object.
8) Perform blob analysis on thresholded image
9) After that box can be drawn around that image.
10) Moving object tracked in that box.
The main steps used are optical flow and thresholding, median filter and blob analysis.

III. THE TRACKING ALGORITHM

The proposed algorithm is presented here. Before any operation a scene should be selected from a static camera. Our test movies are selected from urban surveillance videos. Some pre-processing operations have to be done for making the scene ready to process. Due to the camera's auto white balance and the effect of sudden environment intensity changes, the mean of every frame is calculated on gray-scale format. The optical flow estimation is the essential part of the algorithm which is executed next. Filtering speckle, impulse and general external noises induced due to weather conditions is one of the most important sections of the procedure. Median filter is used in our framework. During filtering operation, some holes are created in the frames. To fill these holes and prevent detection mistakes morphological closing is implemented.

A. Optical flow

Optical flow or optic flow is the pattern of apparent motion of vehicles, surfaces, and edges in a visual scene caused by the relative motion between an observer (an eye or a camera) and the scene.[2][3]. The concept of optical flow was first studied in the 1940s and ultimately published by American psychologist James J. Gibson[4] as part of his theory of affordance. Optical flow techniques such as motion detection, object segmentation, time-to-collision and focus of expansion calculations, motion compensated encoding, and stereo disparity measurement utilize this motion of the objects' surfaces and edges[5][6]. Velocity estimation as follows.

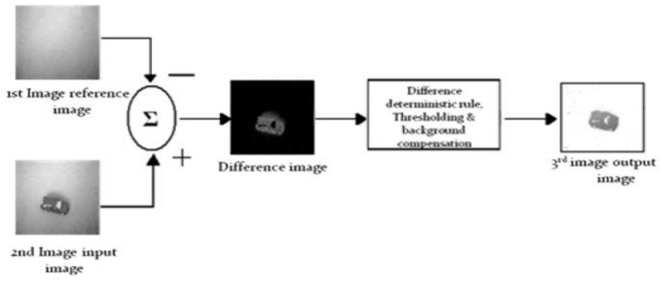

1st Image reference image
−
Σ
+
Difference image
Difference deterministic rule, Thresholding & background compensation
3rd image output image
2nd Image input image

Fig.2 Object detection phase

Estimation of the optical flow:-

Sequences of ordered images allow the estimation of motion as either instantaneous image velocities or discrete image displacements.[6] Fleet and Weiss provide a tutorial introduction to gradient based optical flow [7] John L. Barron, David J. Fleet, and Steven Beauchemin provide a performance analysis of a number of optical flow techniques. It emphasizes the accuracy and density of measurements.[8] The optical flow methods try to calculate the motion between two image frames which are taken at times t and $t + \Delta t$ at every pixel position. These methods are called differential since they are based on local Taylor series approximations of the image signal; that is, they use partial derivatives with respect to the spatial and temporal coordinates. For a 2D+t dimensional case (3D or n-D cases are similar) a voxel at location (x,y,t) with intensity $I(x,y,t)$ will have moved by Δx, Δy and Δt between the two image frames, and the following *image constraint equation* can be given:

$$I(x, y, t) = I(x + \Delta x, y + \Delta y, t + \Delta t)$$

Assuming the movement to be small, the image constraint at $I(x,y,t)$ with Taylor series can be developed to get:

$$I(x+\Delta x, y+\Delta y, t+\Delta t) = I(x,y,t) + \frac{\partial I}{\partial x}\Delta x + \frac{\partial I}{\partial y}\Delta y + \frac{\partial I}{\partial t}\Delta t + \text{Higher order terms}$$

From these equations it follows that:

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

or

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

which results in

$$\frac{\partial I}{\partial x}V_x + \frac{\partial I}{\partial y}V_y + \frac{\partial I}{\partial t} = 0$$

where Vx, Vy are the x and y components of the velocity or optical flow of $I(x, y, t)$ and $\frac{\partial I}{\partial x}$, $\frac{\partial I}{\partial y}$ and $\frac{\partial I}{\partial t}$ are the derivatives of the image at (x, y, t) in the corresponding directions. Ix, Iy and It can be written for the derivatives in the following.
Thus:

$$I_xV_x + I_yV_y = -I_t$$

or

$$\nabla I^T \cdot \vec{V} = -I_t$$

B. Thresholding

Thresholding is the simplest method of image segmentation.
From a grayscale image, thresholding can be used to create binary images.

1) Method

During the thresholding process, individual pixels in an image
are marked as "object" pixels if their value is greater than some threshold value (assuming an object to be brighter than the background) and as "background" pixels otherwise. This
convention is known as threshold Above. Variants include threshold below, which is opposite of threshold above; threshold inside, where a pixel is labeled "object" if its value is between two thresholds; and threshold outside, which is the opposite of threshold inside (Shapiro, et al. 2001:83). Typically, an object pixel is given a value of "1" while a
background pixel is given a value of "0." Finally, a binary image is created by coloring each pixel white or black, depending on a pixel's labels.

2) Threshold selection

The key parameter in the thresholding process is the choice of
the threshold value (or values, as mentioned earlier). Several
different methods for choosing a threshold exist; users can manually choose a threshold value, or a thresholding algorithm can compute a value automatically, which is known as automatic thresholding (Shapiro, et al. 2001:83).
A simple method would be to choose the mean or median value, the
rationale being that if the object pixels are brighter than the background, they should also be brighter than the average. In a
noiseless image with uniform background and object values, the mean or median will work well as the threshold, however, this will generally not be the case. A more sophisticated approach might be to create a histogram of the image pixel
intensities and use the valley point as the threshold. The histogram approach assumes that there is some average value
for the background and object pixels, but that the actual pixel
values have some variation around these average values.
However, this may be computationally expensive, and image
histograms may not have clearly defined valley points, often
making the selection of an accurate threshold difficult. One method that is relatively simple, does not require much specific knowledge of the image, and is robust against image noise, is the following iterative method:

1. An initial threshold (T) is chosen, this can be done randomly or according to any other method desired.
2. The image is segmented into object and background pixels
as described above, creating two sets:
    1) G1 = {f(m,n):f(m,n) > T} (object pixels)
    2) G2 = {f(m,n):f(m,n) ≤ T} (background pixels)
    (note, f(m,n) is the value of the pixel located in the m th
    column, nth row)
3. The average of each set is computed.
    1) m1 = average value of G1
    2) m2 = average value of G2
4. A new threshold is created that is the average of m1 and m2
    1) T' = (m1 + m2)/2
5. Go back to step two, now using the new threshold computed in step four, keep repeating until the new threshold matches the one before it (i.e. until convergence has been reached).

This iterative algorithm is a special one-dimensional case of
the k-means clustering algorithm, which has been proven to converge at a local minimum—meaning that a different initial threshold may give a different final result.

A. Median Filtering

In signal processing, it is often desirable to be able to perform
some kind of noise reduction on an image or signal. The median filter is a nonlinear digital filtering technique, often used to remove noise. Such noise reduction is a typical preprocessing step to improve the results of later processing (for example, edge detection on an image). Median filtering is very widely used in digital image processing because, under certain conditions, it preserves edges while removing noise.
The main idea of the median filter is to run through the signal
entry by entry, replacing each entry with the median of
neighbouring entries. The pattern of neighbours is called the
"window", which slides, entry by entry, over the entire signal.
For 1D signals, the most obvious window is just the first few
preceding and following entries, whereas for 2D (or higher dimensional) signals such as images, more complex window
patterns are possible (such as "box" or "cross" patterns). Note
that if the window has an odd number of entries, then the median is simple to define: it is just the middle value after all
the entries in the window are sorted numerically. For an even
number of entries, there is more than one possible median, see median for more details

B. Noise filtering

Motion digital images are often interfered by a variety of noise distributions dependent on the prevalent conditions. The observed noise can be modeled either as additive white, impulsive, signal dependent or a combination of them. Some of these noise distributions are very annoying when are involved in intensity changes in video frames. They randomly and sparsely corrupt pixels to two intensity levels: relative high or relative low, when compared to its neighboring pixels. Therefore, the need emerges for implementing smoothing techniques that are able to treat different kinds of noise. Furthermore, a noise-free version of the corrupted image or sequence required by adaptive filtering algorithms during the training procedure is not always available. Moreover, it is well known that the main objectives of image filtering algorithms are: (a) the suppression of noise in homogeneous regions, (b) the preservation of edges (spatial or temporal) and (c)the removal of impulses (of constant and/or random value) [17]. A class of filters that fulfills these requirements is the so called signal filters. Standard median (SM) is a paradigm of this class. Median filter, as its name implies, replaces the value of a pixel by the median of the gray levels in the neighborhood of that pixel. Median filters are quite popular because, for certain types of random noise, they provide excellent noise-reduction capabilities, with considerably less blurring than linear smoothing filters of similar size [18].

C. *Image segmentation*

To detect an object, image is normally segmented into blobs or regions using some common segmentation techniques such as background subtraction mean shift clustering and graph cuts. Segmented regions are then grouped together to represent an object based on some deterministic rules [19]; in this paper, the optical flow. In tracking algorithm the content of each frame is read and the background is estimated. The unwanted/interested objects are tagged by eliminating the background. Thresholding function is used to convert the gray image to binary so that the objects of interest can be highlighted by fixing a threshold limit.

D. *Blob Analysis*

In the area of computer vision, blob detection refers to visual
modules that are aimed at detecting points and/or regions in the image that differ in properties like brightness or color compared to the surroundings. There are two main classes of
blob detectors differential methods based on derivative expressions and methods based on local extrema in the intensity landscape. With the more recent terminology used in the field, these operators can also be referred to as interest point operators, or alternatively interest region operators (see also interest point detection and corner detection). There are several motivations for studying and developing blob detectors. One main reason is to provide complementary
information about regions, which is not obtained from edge detectors or corner detectors. In early work in the area, blob detection was used to obtain regions of interest for further processing. These regions could signal the presence of objects
or parts of objects in the image domain with application to object recognition and/or object tracking. In other domains, such as histogram analysis, blob descriptors can also be used
for peak detection with application to segmentation. Another
common use of blob descriptors is as main primitives for texture analysis and texture recognition. In more recent work, blob descriptors have found increasingly popular use as interest points for wide baseline stereo matching and to signal
the presence of informative image features for appearance based object recognition based on local image statistics. There is also the related notion of ridge detection to signal the
presence of elongated objects.

IV. RESULT AND DISCUSSION

In this section we show the experimental results using standard datasets from traffic closed-circuit TV (CCTV) to evaluate our system. Before applying optical flow estimation on frames, the image format is converted from RGB to gray (Fig. 3 (a)); because Intensity measurements act well on gray-scale frames. Depends on methodology steps, the proper optical flow estimation (Locus-Kanade or Horn-Schunk) has been applied. Then, the median filter is performed to reduce noise corruptions. The optical flow estimation regardless of mobility, results in high contrast spots which are focused on.
As it shows in Fig. 2 (a) and Fig. 2 (b), in addition to moving parts, there are some motion vectors pointed regions which are detected as objects, like pedestrian crossing at top left corner of the frame. To overcome this problem morphological close will be operated in algorithm to thin-out the parts of the road and fill holes in the blobs (Fig. 2 (c)).

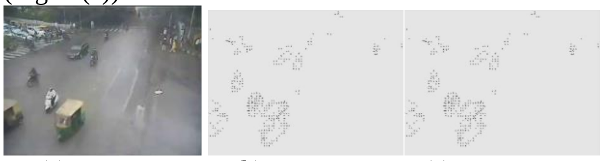
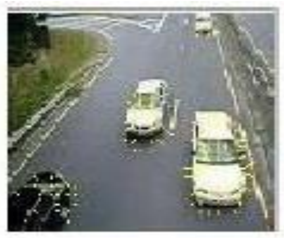
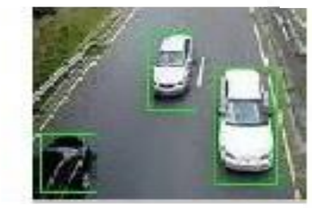

(a) (b) (c)

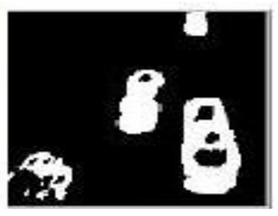
Fig. 2 A selected frame (a) original video (b) optical flow motion vectors before and (c) after applying morphological close

The algorithm continues with blob analysis to detect specific objects, and remove unwanted blobs based on size and features like motion (Fig. 3 (c)). The blob constraints, depend on their utilizations, are fairly varied. The most important one which we have set is the density, i.e., the amount of pixels that exist in one blob. Another is connectivity coefficient, which illustrate the number of pixels that are connected to each other. The larger the number, the higher the accuracy and the more time consuming to compute. The blobs after this stage are surrounded by boxes and are ready for the next step.
The final point in the algorithm is the tracking of moving objects and counting the number of detected vehicles that is shown above the images (Fig. 3 (d)). Before tracking objects, i.e., vehicles (cars, trucks and motorcycles …), it is needed to ensure whether all of the bounding boxes contain vehicles or not. In our experiments, when the ratio of the sides

Input video file:cars.avi


Optical flow motion vectors


Tracking motion vehicles


binary image with blobs

of the bounding box is above 0.4 (40%), it is classified as a vehicle. Table I depicts the average processing times (in ms) of proposed framework when applied on a 120-frame-video and RGB (120×160) image format using MATLAB.

TABLE I

AVERAGE PROCESSING TIMES (IN MILLISECONDS) OF VARIOUS STAGES WITHIN OUR FRAMEWORK

Table Head	*Stages1*	**Methods**	
		Lucas-Kanade	***Horn-Schunk***
1	Optical flow estimation	5.2	6
2	Filtering+segmentation	1.6	1.6
3	Motion vectors computing	0.44	0.58
4	Tracking+count detected number	11.3	11.3
5	Initializing and finalizing	5,534	6,201
6	**Total**	5,552	6,220

Hence due to relatively large number of iterations, Horn-Schunck method is slower but more accurate. This computing cost also appears in motion vectors calculations. Other stages are identical and have the same time consuming.

V. CONCLUSION

This paper has presented a system for motion object detection and tracking in image sequences from aerial or stationary camera images. Traffic surveillance in the urban environment is one of the most applicable usages of this system. The proposed system employs several methods to detect, filtering, segmentation and tracking objects. We used Horn-Schunk algorithm, as the most suitable method of optical flow estimation, to detect moving objects by the intensity changes of frames. The median filter performance significantly surpasses that of the other filters under study. The morphological close extracted significant features of region shapes from binary images and then blob analysis introduced these shapes to the next step as foregrounds. A great advantage of blob analysis is the low computation cost. Finally, as shown in the experimental result, the system is able to remove unwanted motion object which are not vehicles in the image sequence, using constraints on blob areas. For the future work, ego motion estimation can be employed to compensate the camera shakings or to use this method on aerial videos. An adaptive filter helps this system to improve noise reduction. Also the algorithm may be developed to detect and identify overlapping objects and occlusion or transparencies during object tracking.

REFERENCES

[1] J. L. Barron, D. J. Fleet, and S. S. Beauchemin, "Systems and experiment performance of optical flow techniques," Int. J. Comput. Vis., vol.12, 1, pp. 43–77, Feb. 1994.
[2] J. Xiao, H. Cheng, H. Sawhney, C. Rao, and M. Isnardi, "Bilateral filtering- based optical flow estimation with occlusion detection," ECCV, 2006, pp. 211–224.
[3] T. Brox, A. Bruhn, N. Papenberg, and J. Weickert, "High accuracy optical flow estimation based on a theory for warping," ECCV, 2004, pp. 25–36.
[4] C. L. Zitnick, N. Jojic, and S. B. Kang, "Consistent segmentation for optical flow estimation," IEEE ICCV, 2005, pp. 1308–1315.
[5] M. J. Black, "The robust estimation of multiple motions: Parametric and piecewise-smooth flow fields," Comput. Vis. Image Underst., vol.63, no. 1, pp. 75–104, Jan. 1996.
[6] Rakshit, S.; Anderson, C.H., "Computation of optical flow using basis functions," Image Processing, IEEE Transactions on, vol.6, no.9, pp.1246, 1254, Sep 1997.
[7] Zhan, Wei, and Xiaolong Ji. "Algorithm Research on Moving Vehicles Detection," Procedia Engineering, vol. 15, 2011, pp. 5483-5487.
[8] Ballard, Dana H. "Generalizing the Hough transform to detect rbitrary shapes," Pattern recognition, vol. 13, no. 2, 1981, pp. 111-122.
[9] Illingworth, John, and Josef Kittler. "A survey of the Hough transform," Computer vision, graphics, and image processing, vol. 44, no. 1, 1988, pp. 87-116.
[10] Shi, Xuejie. "Research on Moving Object Detection Based on Optical Flow Mechanism." University of Science and Technology of China, 2010, pp. 6-14.
[11] Qiaona Pei. "Moving Objects Detection and Tracking Technology based Optical Flow". North China University of Technology, 2009, pp. 11-14.
[12] Lucas B. D., Kanade T. "An Iterative Image Registration Technique with an Application to Stereo Vision," DARPA Image Understanding Workshop, 1981, pp. 121–130.