International Journal of Electronics, Communication & Soft Computing Science and Engineering
ISSN: 2277-9477, Volume 2, Issue 1

Moving Object Tracking in Video Using MATLAB

Bhavana C. Bendale, Prof. Anil R. Karwankar

***Abstract* —In this paper a method is described for tracking moving objects from a sequence of video frame. This method is implemented by using optical flow (Horn-Schunck) in matlab simulink. It has a variety of uses, some of which are: human-computer interaction, security and surveillance, video communication and compression, augmented reality, traffic control, medical imaging and video editing.**

***Keywords*— augmented, surveillance, medical imaging, MATLAB simulnik.**

I. INTRODUCTION

The objective of this project is to identify and track a moving object within a video sequence. The tracking of the object 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 consist 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 "Object 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 object motion using either the Horn-Schunck or the Lucas-Kanade. The velocity estimation can be either between two images or between current frame and N^{th} 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 object 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 "regionprops" 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. DESIGN AND IMPLEMENTATION

In this Simulink model, there are couple 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 object may have many small boundary boxes due to the optical detection on different part of the moving object. In order to better keep track of the moving object, 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.

III. 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 N^{th} 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.
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.

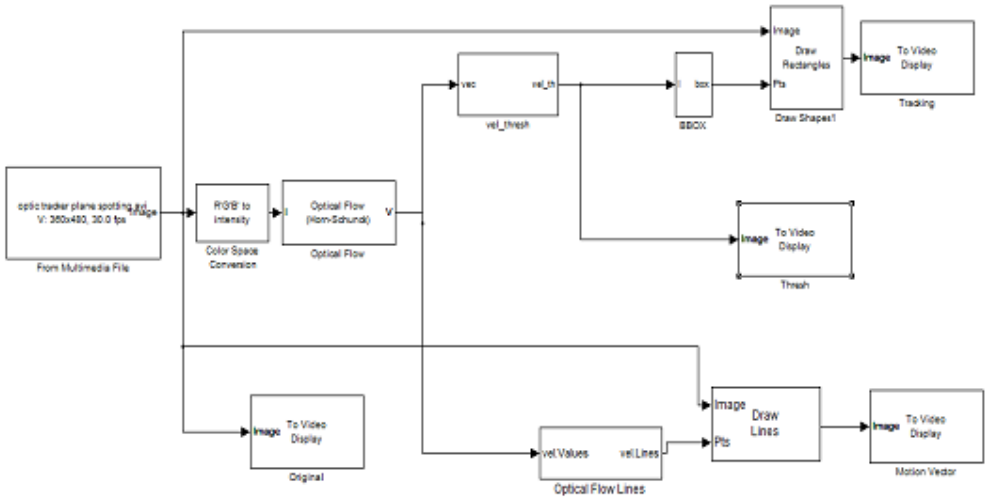

Fig. 1 Simulink Block Diagram for Tracking Moving Objects Using Optical Flow

A. Optical flow

Optical flow or optic flow is the pattern of apparent motion of objects, 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.

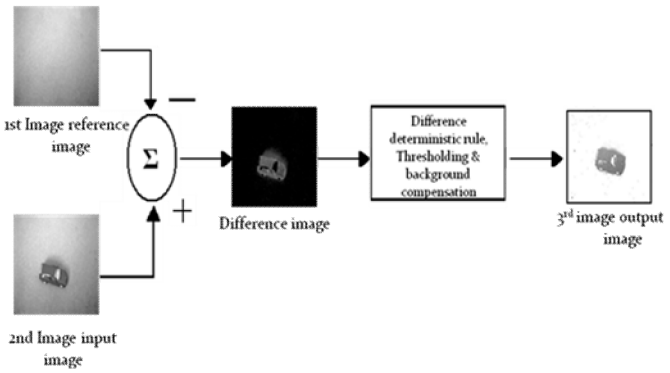

Fig.2 Object detection phase

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 voxel 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:

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

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 V_x, V_y 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. I_x, I_y and I_t can be written for the derivatives in the following.

Thus:

$$I_x V_x + I_y V_y = -I_t$$

or

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

B. Thresholiding

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 asthreshold 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) $G_1 = \{f(m,n):f(m,n) > T\}$ (object pixels)
    2) $G_2 = \{f(m,n):f(m,n) \leq T\}$ (background pixels) (note, f(m,n) is the value of the pixel located in the m^{th} column, n^{th} row)
3. The average of each set is computed.
    1) m_1 = average value of G_1
    2) m_2 = average value of G_2
4. A new threshold is created that is the average of m_1 and m_2
    1) $T' = (m_1 + m_2)/2$
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.

C. 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 pre-processing 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.

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 surrounding. 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

Input video file: Walk.avi

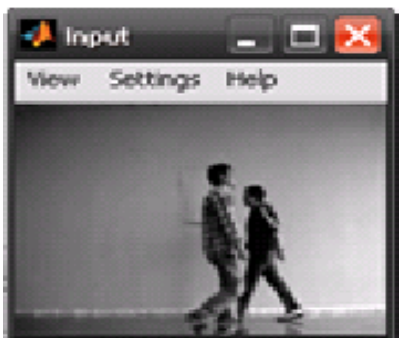

Fig 3. Original input video

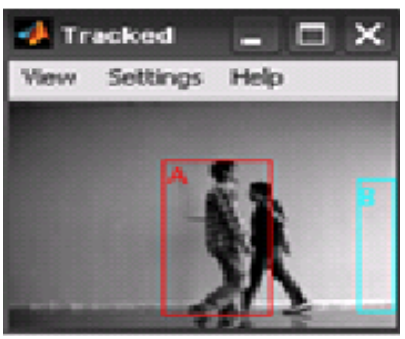

Fig. 4 Thresholding for blob detection

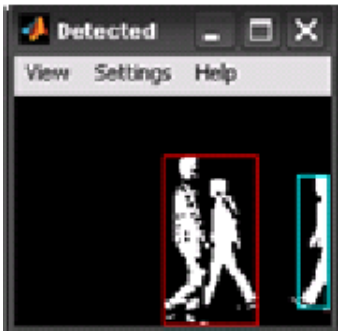

Fig. 5 Tracking

This is tracking without background extraction. Because while extracting background from video frame if there are small moving things in that frame they form a blob in thresholding which create confusion in case of tracking that blob as they aren't of any use that can be reduced here.

V. FUTURE WORK

In future this method can be modified to differentiate different class objects in real time video or this can be used to perform obstacle avoidance for robots or cars etc.

REFERENCES

[1] LeFloch, D. Real-time people counting system using video camera. Master's thesis, Gjoevik University College, Universite de Bourgogne, 2007.
[2] Object Tracking: A Survey, Alper Yilmaz, Omar Javed, Mubarak Shah
[3] Brown, L. M. View independent vehicle/person classification. Technical report, Proceedings of the ACM 2nd international workshop on Video surveillance & sensor networks, pages 114-123, 2004.
[4] Masamitsu Tsuchiya, H. F. Evaluating feature importance of object classification in visual surveillance. Technical report, The 18th International Conference on Pattern Recognition (ICPR'06), Vol 2, pages 978-981, 2006.
[5] Ying-Li Tian, M. L. & Hampapur, A. Robust and efficient foreground analysis for real-time video surveillance. Technical report, Proceedings of the 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05), Vol 1, pages 1182-1187, 2004.
[6] Omar Javed, M. S. Tracking and object classification for automated surveillance. Technical report, Proceedings of the 7th European Conference on Computer Vision- Part IV, pages 343-357, 2002.
[7] Schofield A.J, S. T. & P.A, M. A ram based neural network approach to people counting. Technical report, Image Processing and its Applications, 1995., Fifth International Conference, pages 652-656, July 1996.

AUTHOR'S PROFILE

Bhavana C. Bendale
Department of Electronics and Telecommunication Engineering
J. T. Mahajan College of Engineering,
Faizpur, Maharashtra, India.
bendalebhavana@gmail.com

Prof. Anil R. Karwankar
Department of Electronics and Telecommunication Engineering
Govt College of Engineering,
Aurangabad, Maharashtra, India