DESIGN AND IMPLEMENTATION OF AN OPTICAL FLOW-BASED AUTONOMOUS VIDEO SURVEILLANCE SYSTEM

A. Fonseca, L. Mayron, D. Socek, and O. Marques.
Florida Atlantic University
777 Glades Rd., Boca Raton, FL 33428 USA
afonsec3@fau.edu, lmayron@fau.edu, dsocek@brain.math.fau.edu, omarques@fau.edu

ABSTRACT
Autonomous video surveillance systems typically consist of several functional modules working in concert. These modules perform specialized tasks including motion detection, separation of the foreground and background, depth estimation, object tracking, feature estimation, and behavioral analysis. Computational overhead and redundancy may result from designing each module individually, as each module may incorporate different variety of techniques and algorithms. This paper presents the design of a surveillance system that uses an optical flow algorithm throughout. We consider the capabilities, solutions, and limitations of this design. Additionally, an evaluation of the performance of optical flow in specific situations, such as depth estimation, rigid and non-rigid classification, segmentation, and tracking, is provided. The main contribution of this work is a new system-level architecture based on a single key algorithm (optical flow) for the entire video surveillance system.

KEY WORDS
Optical flow, video surveillance, segmentation, tracking, depth estimation, feature extraction

1. Introduction

Video surveillance systems [1, 13, 14] are often designed as modular systems composed of a variety of functional blocks such as motion detection, object tracking, depth estimation and object behavioral analysis [1, 4-7]. Adaptive systems, Kalman Filters, neural networks, and several image and video processing algorithms [2, 5] are being used without consideration of the impact of their computational performance on the entire system. Components are connected in a serial configuration where the intermediate results of each block are not shared [9]. We focus on creating a more integrated video surveillance system based on one algorithm, optical flow. We demonstrate that it is feasible to share information between processing blocks, given a new alternative into the system level design scope.

Once optical flow is computed the measurement of image velocity can be used for a wide variety of tasks ranging from passive scene interpretation to autonomous, active exploration [8]. Optical flow calculates disparities and depth estimation. Segmentation algorithms based on optical flow; objects matching and tracking are some examples of the optical flow potential [2, 3]. Therefore, optical flow is a strong candidate to demonstrate the performance for a surveillance system based on a single algorithm (as opposed to a combination of heterogeneous functional blocks). Performance is measured with different videos under different circumstances; objects classification, occlusion analysis and depth estimation will be part of testing. Although other algorithms could have been used in lieu of optical flow, the main contribution to the state-of-the-art is use of a single technique to reduce hardware resources and processing time while maintaining acceptable surveillance performance and quality.

Restricting our system to optical flow information presents new alternatives for handling segmentation, depth estimation, tracking, and object classification. Implementation is realized using a single camera. Despite this, the same optical flow method was able to estimate depth. We emphasize that the applied framework is the optical flow method, whereas the ultimate objective is the implementation of an entire video surveillance system.

2. Proposed Solution

This section describes the design and implementation of an autonomous video surveillance system based on optical flow calculations.

Figure 1 shows a high-level block diagram of the proposed system. The *optical flow (OF) calculation* block is highlighted. Its results will drive several other blocks. of each block follows.

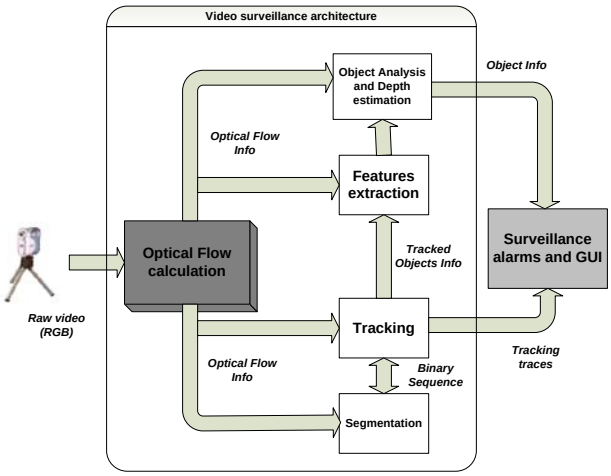


Figure 1 System architecture

Information is shared at the system level. Optical flow supplies information for every module. Data is also shared among the various sub-blocks. This system architecture is designed for reducing the number of implemented algorithms and increasing the sharing of data among modules.

2.1 Optical Flow Calculation

The most common optical flow algorithms are listed below:

- Lucas-Kanade (Local method) [10]
- Horn-Schunck (Global method) [9]
- Optical flow using window pixels correlation [11]

According to the argument presented in [8], the Lucas-Kanade algorithm was selected as the most appropriate algorithm for this implementation. In summary, it provides the lowest processing time while maintaining acceptable results.

2.2 Segmentation

The *Segmentation* block uses the results of the optical flow calculations to determine which pixels in the frame belong to the foreground and which to the background. The threshold is a dynamic parameter and its value may change from one frame to the next due to several factors including weather, illumination and camera settings.

Figure 2 shows the average of optical flow vector length per frame. Frames 1 through 30 consist only of the background. As a results the average optical flow vector length is very low (the average is 0.04). After frame 30 moving objects begin to appear and the optical flow vector length increases accordingly. This observation is used to separate background and foreground optical flow vectors using thresholding.

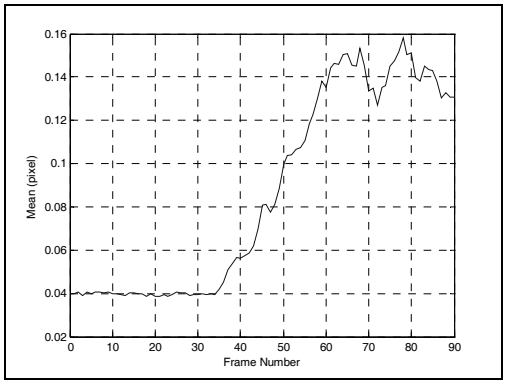


Figure 2 Average optical flow vector magnitude, per frame

The main challenge within the segmentation block is the handling of small optical flow vectors. These may be associated with either background noise or with slow-moving, but relevant objects. We solved this problem in the following way:

1. Using a few initial frames with only background information, we calculate an initial threshold that can be used to suppress the optical flow noise. Threshold T is calculated assuming Gaussian noise with mean and standard deviation of the OF vector sizes (μ_B, σ_B), and a percentage of noise elimination P (usually greater than 99%). At this point it is relevant to mention that T is not an efficient value; it is higher than normal in order to eliminate background noise; affecting short-length foreground object vectors and reducing object contour accuracy.

2. Although several foreground object vectors were eliminated we now have the number of real moving objects N for tracking. We reduce T to $T_1 = T - \tau$ and then we count the number of objects N_1. If $N_1 = N$ then T_1 can be reduced again.

3. This process is repeated until the number of objects N_i at iteration i differs from N.

2.3 Tracking

In our system each object (as determined by the *Segmentation* block) is represented as a bounding box and the coordinates of the object's centroid. Optical flow vectors are used as an object feature for tracking. Objects' trajectories are represented by centroid displacement.

The proposed tracking method is based on the optical flow tracking algorithm described in [12]. The main steps involved in determining where an object moved to are as follows:

1. Standard deviation σ and mean μ of the optical flow vectors surrounded by a bounding box in frame *k – 1* are calculated and stored.
2. The bounding box in the frame *k – 1* is shifted by the mean flow of that frame. This shifted window is refered to as the *prediction window.*
3. Objects whose centroid is surrounded by the prediction window in frame *k* are going be part of the tracked object.
4. Bounding box, standard deviation σ and mean μ are calculated again for frame *k*.
5. Steps 2-4 are repeated until end of the video sequence.

Internal data structures keep track of two types of objects: the ones which have been tracked for more than *p* consecutive frames (which are called *ActiveObjects*) and the ones which have not (*BufferObjects*). These data structures are updated on a frame-by-frame basis, instantiating new objects and promoting objects from the *BufferObjects* to the *ActiveObjects* category. The value of *p* must be chosen so that noise-like objects will not be tracked. For visualization purposes, objects currently being tracked are enclosed by a bounding box. Their trajectory is painted on the screen, as shown in see Figure 4.

2.4 Occlusion and objects without motion

Occlusion is handled entirely within the tracking process:

1. Recall locations and objects that have disappeared and are within in the borders.
2. Create an alert when a new object has appeared and it is not within the border.
3. Matching optical flow features and the aspect ratio of previous lost objects with the new no-borders objects.

If an object was close to the place where previous object disappear then it will be treated as non-moving instead of occluded. Still objects could not be tracked due of the motion-requirements of optical flow. Thus, stopped objects are handled in the tracking block. This extra computation does not affecting the overall processing time of the tracking subsystem.

2.5 Feature extraction

Feature extraction has been implemented using only optical flow-derived features, consistent with the design philosophy of this system. Consequently, we use the following features to represent each object:

- The mean value and the standard deviation of the x and y components of all the optical flow vectors associated with the object, per frame
- The number of optical flow vectors per object ,per frame
- The aspect ratio of the object's bounding box, per frame

The modular nature of our architecture enables other, richer features (e.g., shape-, texture, and/or color-based) to be incorporated in the future.

2.6 Object analysis

A typical step in behavior analysis is the classification of objects as either rigid or non-rigid. This can also be realized using optical flow, with the assumption that optical flow vectors contained within the bounding box of a rigid object are likely to be parallel to the object's theoretical displacement vector, whereas for non-rigid objects this will not hold.

We calculate the mean of the horizontal and vertical optical flow vector to help us determine, not only know the direction of an object, but also the oscillatory behavior of non-rigid objects. We also calculate the standard deviation of the optical flow vectors to determine how far the vector components of the objects are from the theoretical object displacement vector: the smaller the standard deviation, the more likely to be a rigid object.

2.7 Relative depth estimation

It has been observed that the number of optical flow vectors per object diminishes when the object moves away from the camera. This, in conjunction with the mean x and y optical flow values per object, is used to estimate the 3D trajectory of an object. Although calibration parameters are not taken into account; this first approach will show that this relative depth estimation is useful for:

- Handling groups of objects merging together
- Raising an alarm if an object is within a non-permitted zone
- Estimating an object's trajectory

The process of relative depth estimation is depicted as follows:

- The number of pixels and mean value of the *x* and *y* components of all the optical flow vectors associated with the object is calculated for each frame.
- Digital filtering is applied to the previous frame's information for every frame in which the object appears to remove non-rigid oscillations.
- Displacement information is calculated by integrating object velocities. Different view plots and gray scale depth maps are stored for future analysis.

3. Experiments and Results

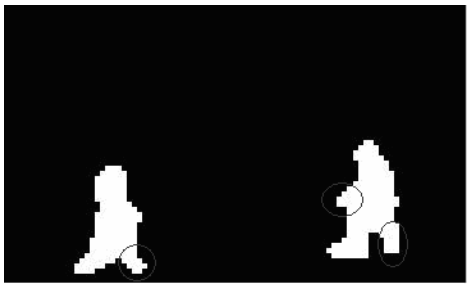
The system is implemented in MATLAB. The sequences were taken in different scenarios with only one camera. Each frame was captured at a 320 by 240 pixel resolution.

3.1 Segmentation

Figure 3 shows optical flow vectors (a) (the Lucas-Kanade algorithm with a window of 4x4 pixels), original segmentation (b), and corrected segmentation (c). Notice that parts of arms and legs are ultimately recovered by the corrected segmentation method.

(a) Optical flow

(b) Original Segmentation



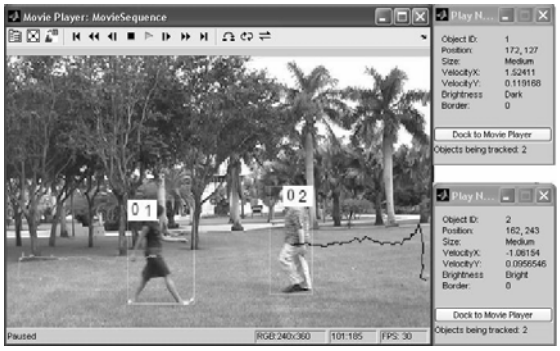
(c) Corrected segmentation

Figure 3 Segmentation process

3.2 Tracking system

We tested the tracking system by measuring successfully-recovered occlutions. Each object in our system is assigned a unique ID. Successful occlusion identification occurs when this ID is correctly maintained before and after occlusion.

Figure 4 shows that a given object retains its ID before (a) and after (c) occlusion. Figure 2(b) indicates that the object assigned ID 2 was occluded. Relevant information for each object (object ID, position, size, velocity, brightness, and border) is also displayed.



(a) Before Occlusion

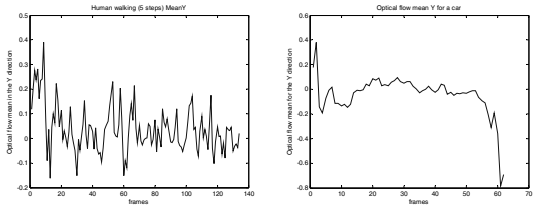
(b) Occlusion

(c) After Occlusion

Figure 4 Object tracking in the presence of occlusion

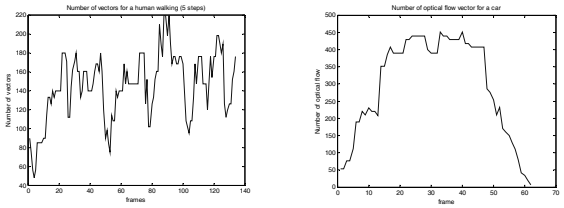
3.3 Object classification

Figures 5 and 6 show the rigid and non-rigid behavior of a human (a) and car (b) for the vertical component and number of pixels of optical flow. Notice the oscillatory behavior of human signals which is reflected in standard deviation measures.



(a) **(b)**

Figure 5 Mean vertical components for a human (a) and car (b)



(a) **(b)**

Figure 6 Mean number of optical flow vectors for a human (a) and car (b)

Table 1 Object Classification (Standard deviation)

	TP rate	FP rate	Precision	Recall
Human	1	0.125	0.909	1
Car	0.875	0	1	0.875

Table 2 Object Classification (Standard deviation and Aspect ratio)

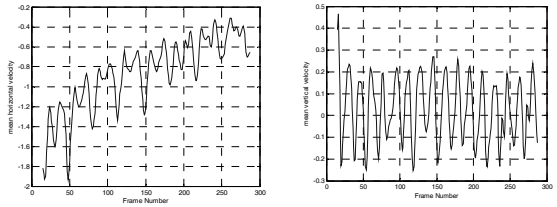
	TP rate	FP rate	Precision	Recall
Human	0.9	0	1	0.9
Car	1	0.1	0.889	1

Fifty human and car samples were taken and classified by using J48 tree classifier in WEKA. Tables 1 and 2 show the results of object classification using standard deviation alone and standard deviation with aspect ratio measures.

3.4 Relative depth estimation

Figure 7 Object trajectory for depth estimation

To determine the relative depth estimation of the trajectory shown in Figure 7 we use information from Figures 8 and 9. Information from Figure 8 lets us describe the vertical and horizontal trajectory based on the average of optical flow vertical and horizontal components of the objects, respectively.



(a) **(b)**

Figure 8 Average of OF Vertical and Horizontal components per frame

Depth trajectory is calculated using the information shown in Figure 9. The number of optical flow vectors for the object determines how close is the objects to the camera. The further away the object is, the greater the number of optical flow vectors.

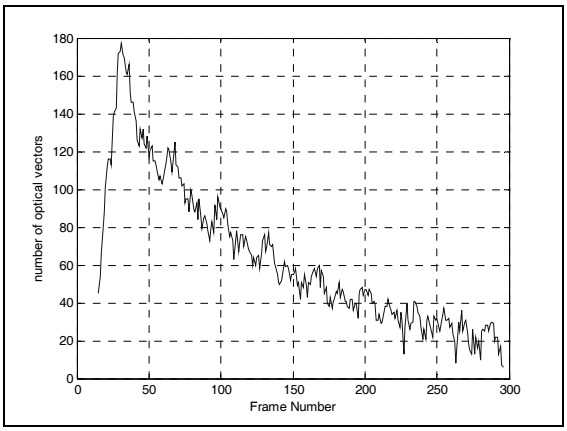


Figure 9 OF number of pixels per frame

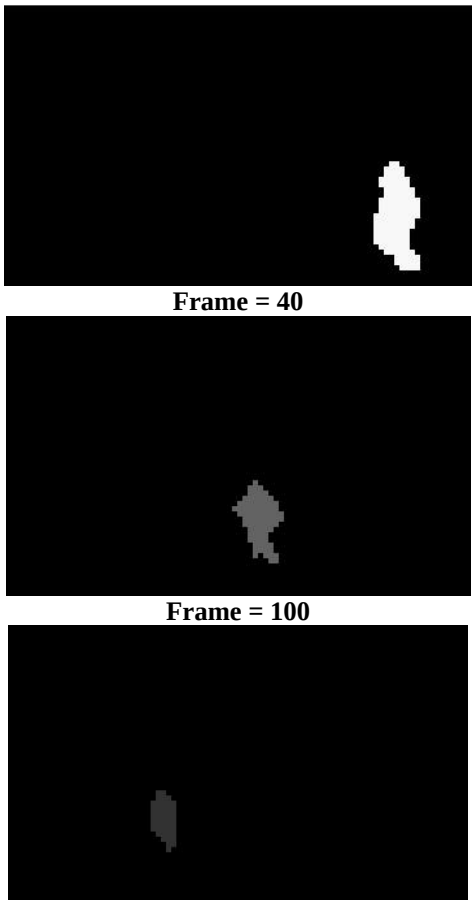
Prior information is used to estimate the relative depth ,as shown in Figure 10 (a grayscale depth plot).

4. Conclusion and future work

The most significant limitation when using optical flow for video surveillance is the case of non-moving objects. These objects do not generate optical flow. If an object stops it will be temporarily lost. However, our system does detect the case when an object is no longer tracked but was not near the edge of the frame when it "disappeared".

We have shown that a surveillance system can be designed by using a single technique (optical flow) with acceptable performance. An algorithm for reconstructing segmented images is proposed for solving optical flow thresholding. Improvements such as faster algorithms and consecutive frames subtraction for reducing segmentation processing is part will be investigated in the future

.



Frame = 40

Frame = 100

Frame = 160

Figure 10 Relative Depth Map

References

[1] R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, A system for video surveillance and monitoring: VSAM final report, *Robotics Inst., CMU-RI-TR-00-12*. 2000.

[2] W. Hu, T. Tan, L. Wang, and S. Maybank, Survey on Visual Surveillance of Object Motion and Behaviors, *IEEE Transactions on systems and cybernetics*, *34*, 2004, 334-353.

[3] Gilad Avid, Determining three-dimensional motion and structure from optical flow generated by several moving objects, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *7*, 1985, 384-401.

[4] T. Kanade, R. Collins, A. Lipton, P. Burt, and L. Wixson, Advances in cooperative multisensor video surveillance, In *Proceedings of the 1998 DARPA Image Understanding Workshop*, *1*, 1998, 3–24.

[5] A. Yilmaz, O. Javed, and M. Shah, Object Tracking: A Survey, *ACM computing surveys*, 38(4), 2006, 1-45.

[6] D. Duque, H. Santos, and P. Cortez, The OBSERVER: An Intelligent and Automated Video Surveillance System, *Proceedings of the International Conference on Image Analysis and Recognition*, Povoa de Varzim, Portugal, 2006, 898-909.

[7] I. Haritaoglu, D. Harwood, and L. Davis, W4: Real-time surveillance of people and their activities, *IEEE Trans. Pattern Anal. Mach. Intell.*, *22*, 2000, 809–830.

[8] J. Barron, D. Fleet, and S. Beauchemin, Performance of Optical Flow Techniques, *Int. J. Comput.Vis*, *12*, 1994, 42–77.

[9] B. Horn and B. Schunck, Determining Optical Flow, *Artificial Intelligence*, *17*, 1981, 185-203.

[10] B. Lucas, and T. Kanade, An Iterative Image Registration Technique with an Application to Stereo Vision, *Proc. DARPA IU Workshop*, 1981, 121-130.

[11] W. Pratt, Correlation Techniques of Image Registration, *IEEE Trans. Aerospace and electronic system*, *10*(3), 1974, 353-358.

[12] H. Tsutsui, J. Miura, and Y. Shirai, Optical flow-based person tracking by multiple cameras, in *Proc. IEEE Conf. Multisensor Fusion and Integration in Intelligent Systems*, 2001, 91–96.

[13] Y. Ivanov, C. Stauffer, A. Bobick, W. Grimson, Video Surveillance of Interactions, *Video Surveillance IEEE second workshop on*, 1999, 82-89.

[14] G.L. Foresti, A Real-Time System for Video Surveillance of Unattended Outdoor Environments, IEEE *Trans. Circuits and Systems for Video Technology*, *8*(6), 1998, 697-704.