Robust Camera Pan and Zoom Change Detection Using Optical Flow

Vishnu V. Makkapati
Philips Research Asia - Bangalore
Philips Innovation Campus, Philips Electronics India Ltd.
Manyata Tech Park, Nagavara, Bangalore - 560 045, India
Email: vishnu.makkapati@philips.com

***Abstract*—Detection of camera motion is essential for automated video analysis. Changes in camera pan and zoom are two important events to be identified. This paper proposes optical flow based techniques to identify these events. The scheme detects pan change by finding the number of flow vectors that correspond to the dominant orientation. An orientation histogram has been proposed to quantify the direction of the flow vectors. Zoom-out and zoom-in are identified using the convergence and divergence of flow vectors respectively. A distance based method has been presented to identify the presence and nature of zoom. Numerical results presented for various data sets show that the proposed scheme is very robust.**

I. Introduction

Video surveillance systems are being deployed at many critical locations to monitor unintended events. The availability of low-cost cameras and computing power made these systems affordable and feasible to be used. Analyzing the voluminous data acquired by surveillance cameras is becoming increasingly important. Detection of camera motion is critical for any automated video analysis system. Pan and zoom change are two important camera operations to be identified.

Traditional motion detection algorithms cannot be applied directly to this problem since these do not distinguish between the movement of a camera and an object(s) in the scene. The problem becomes more complicated when the camera is installed on a vibrating platform. The movement of the platform itself should be distinguished from that of the camera.

A method for camera dysfunction detection has been proposed in [1]. This scheme detects pan change by using *block matching* which maximizes the normalized correlation between a reference accumulator and the current accumulator. The parameters of translation are identified and pan change is assumed to have happened if they exceed a certain threshold.

In the context of scene change detection, several methods that work either in the *image domain* or the *compressed domain* (e.g., MPEG) have been proposed. The method discussed in [2] characterizes camera motion using the six-parameter affine model. Zoom-in and zoom-out are commonly detected by testing for existence of a *Focus of Expansion* or a *Focus of Contraction* [3].

This work was performed when the author was employed with Honeywell Technology Solutions Lab, India

The previous methods, in general, estimate the parameters of the respective models using least-square principle. Hence they may not perform well when large number of objects in the scene move. In this paper, a robust method based on optical flow is proposed to detect change of pan and zoom. Section II gives an overview of optical flow. The proposed scheme is described in III. Performance of the scheme is evaluated in section IV. Finally, conclusions are drawn in section V.

II. Optical Flow

If a camera or an object(s) in the scene moves, the resulting apparent motion in the image is called *optical flow*. Optical flow describes the speed and direction of motion of feature points in the image. Assuming that a pixel at location (x, y, t) with intensity $I(x, y, t)$ has moved by δx, δy, and δt between two frames, the image constraint equation is given as

$$I(x, y, t) = I(x + \delta x, y + \delta y, t + \delta t) \tag{1}$$

If the movement is small enough, the Taylor series expansion may be written as

$$I(x+\delta x, y+\delta y, t+\delta t) = I(x, y, t) + \frac{\partial I}{\partial x}\delta x + \frac{\partial I}{\partial y}\delta y + \frac{\partial I}{\partial t}\delta t \tag{2}$$

where the higher order terms are ignored. Equations (1) and (2) give

$$\frac{\partial I}{\partial x}\frac{\delta x}{\delta t} + \frac{\partial I}{\partial y}\frac{\delta y}{\delta t} + \frac{\partial I}{\partial t}\frac{\delta t}{\delta t} = 0 \tag{3}$$

which is commonly written as

$$I_x V_x + I_y V_y + I_t = 0 \tag{4}$$

where $I_x = \frac{\partial I}{\partial x}$, $I_y = \frac{\partial I}{\partial y}$, $I_t = \frac{\partial I}{\partial t}$, $V_x = \frac{\delta x}{\delta t}$, and $V_y = \frac{\delta y}{\delta t}$. This is an equation in three unknowns and cannot be solved directly. Another set of equations is required to find a solution and these are usually given by additional constraints.

The computation of optical flow requires the corresponding feature points in successive frames and several methods have been proposed for this purpose [5]. A common approach is to use the similarity between image patches surrounding the individual feature points. Two measures of similarity namely *sum of squared differences* and *cross-correlation* are often used for this purpose. One measure is the sum of squared

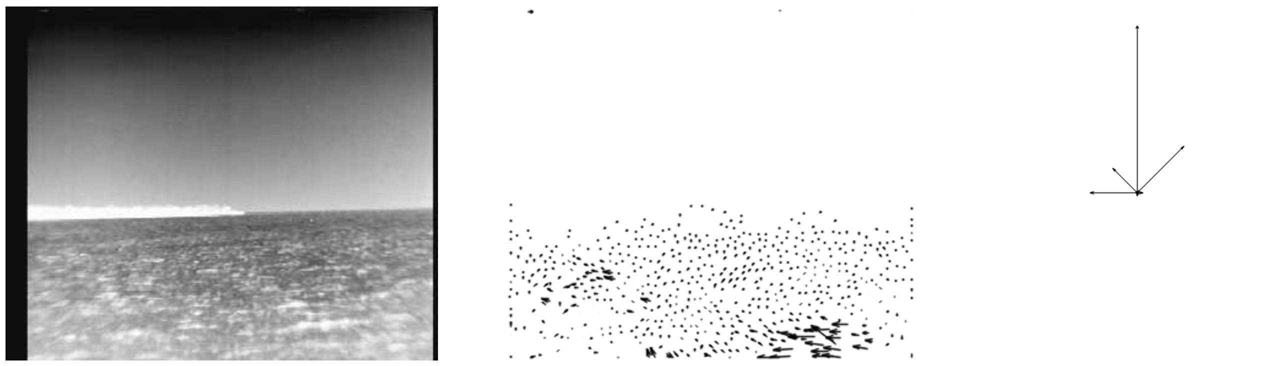

(a) Frame of an image sequence [4] (b) Optical flow vectors (c) Orientation Histogram

Fig. 1. Random optical flow vectors generated due to waves in a sea

differences (SSD) between an image patch at location (x, y) at time t and the candidate locations $(x + \delta x, y + \delta y)$ where that image patch could have moved after time δt. The solution to this problem is to find the displacement in image plane $(\delta x, \delta y)$ which minimizes the SSD criterion:

$$\text{SSD}(\delta x, \delta y) = \sum_{x,y} \left(I(x, y, t) - I(x + \delta x, y + \delta y, t + \delta t)\right)^2 \tag{5}$$

where the summation ranges over the image patch centered at the feature point of interest. The other measure of similarity is to maximize the cross-correlation between the patches in the frames expressed as

$$\text{Cor}(\delta x, \delta y) = \sum_{x,y} I(x, y, t) \cdot I(x + \delta x, y + \delta y, t + \delta t) \tag{6}$$

However, optical flow is not uniquely determined by the local information. The fact that optical flow along the direction of the brightness pattern cannot be determined is termed as *aperture problem*. Hence, complete optical flow is best determined at the corners.

Lucas and Kanade proposed a non-iterative method [6], [7] to solve equation(3). It is a first order, local differential method and assumes a locally constant flow. It is found to be most reliable among optical flow techniques [8]. This method assumes that the flow (V_x, V_y) is constant in a small window of dimension $m \times m$ (where $m > 1$) which is centered at pixel location (x, y). If the pixels are numbered sequentially as $1 \ldots n$ then the following equation is obtained:

$$\begin{bmatrix} I_{x,1} & I_{y,1} \\ I_{x,2} & I_{y,2} \\ \vdots & \vdots \\ I_{x,n} & I_{y,n} \end{bmatrix} \begin{bmatrix} V_x \\ V_y \end{bmatrix} = \begin{bmatrix} -I_{t,1} \\ -I_{t,2} \\ \vdots \\ -I_{t,n} \end{bmatrix} \tag{7}$$

This results in more than two equations for two unknowns. This is an over-determined system and may be solved using least squares method. By using this method, optical flow can be calculated using the derivatives of an image in all three dimensions (i.e., x, y, and t axes).

A weighting function is generally employed to give more importance to the center pixel of the window, with Gaussian function being the most preferred. A coarse-to-fine iterative version of this method is used for image registration. This method first computes the spatial derivatives at a coarse scale in *scale-space*, then warps one of the images by the computed deformation, and then calculates the iterative updates at successively finer scales.

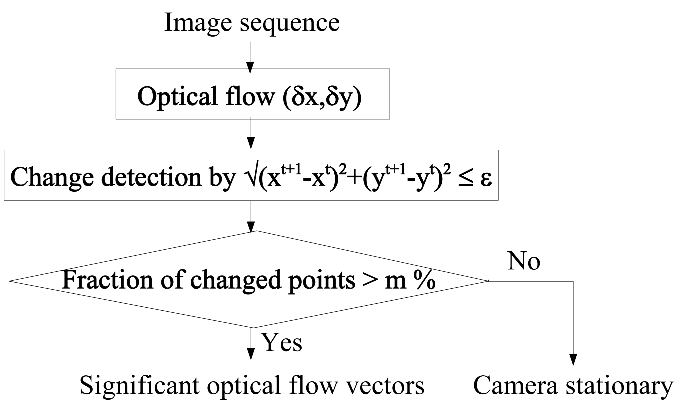


Image sequence
Optical flow (δx,δy)
Change detection by √(x^{t+1}-x^t)^2+(y^{t+1}-y^t)^2 ≤ ε
Fraction of changed points > m %
No
Yes
Significant optical flow vectors
Camera stationary

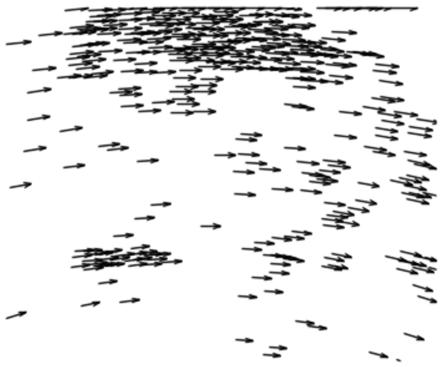
Fig. 2. Flowchart of motion detection algorithm

A characteristic feature of Lucas-Kanade algorithm and that of other local optical flow algorithms is that a very high density of flow vectors is not obtained. The flow information fades out quickly across motion boundaries and inner parts of large homogenous patches show little motion. The primary advantage is the robustness in the presence of noise.

III. Proposed Scheme

In this work the direction (or orientation) of optical flow vectors is used to distinguish the motion of a camera from that of an object(s) in the scene. The scheme assumes that the movements of natural objects such as trees, waves etc. are random. This behavior may often be observed in the motion of a group of humans and animals. However, the motion induced by the camera will be in a particular direction. This fact is illustrated in Fig.1(b), Fig.3(b), and Fig.5(b) representing natural, camera pan change, and camera zoom-in induced motions respectively.

(a) Frame of an image sequence


(b) Optical flow vectors

(c) Orientation Histogram

Fig. 3. Dominant optical flow direction in the presence of pan change

A. Motion Detection

The scheme first determines the feature points (x_i^t, y_i^t) in a frame at time t using Lucas-Kanade method [7]. The matching feature points (x_i^{t+1}, y_i^{t+1}) for these points in the frame at time $t+1$ are obtained using pyramidal-optical flow algorithm [6]. Accuracy and robustness are two key components of any feature tracker. Small integration window provides local sub-pixel accuracy by avoiding *smooth out* of the details contained in the images. Large integration window handles large motions and improves sensitivity of tracking. Therefore a tradeoff exists between local accuracy and robustness when choosing the integration window size. The pyramidal implementation provides an optimal solution to that problem.

The feature points will remain stationary if there are no moving objects in the scene and the camera was stationary. Any feature point identified on a moving object will get displaced across two successive frames. The basic assumption here is that at least m% of the feature points in any scene would lie on stationary objects if there is no movement of the camera and the number of the moving objects in the scene is less. The number of feature points that remain stationary between two successive frames is computed. For this purpose, the Euclidean distance between the matched feature points is checked to be less than a certain tolerance value ϵ.

$$\left|\sqrt{(x_i^{t+1} - x_i^t)^2 + (y_i^{t+1} - y_i^t)^2}\right| \leq \epsilon \tag{8}$$

Each feature point that satisfies this condition is tagged to be stationary. The threshold ϵ also handles any movement that may have been induced due to the vibration of the camera platform. An appropriate value of ϵ can be chosen depending on the application. The value of ϵ may be identified based on the frame rate of the camera and the processing power available. If the frame rate is lower, the feature points defining the flow vectors may be farther apart and a higher ϵ value may be appropriate. This choice also holds when the computing power is less, in which case the input frames may have to be sampled at a higher rate (i.e., successive frames are separated by a larger δt). Fig.2 shows the steps involved in the motion detection algorithm. The scheme then proceeds to the next step where the optical flow field is classified as random or camera-induced.

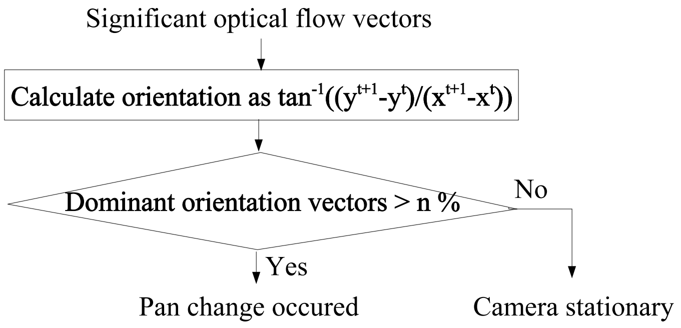

Significant optical flow vectors
Calculate orientation as tan-1((yt+1-yt)/(xt+1-xt))
Dominant orientation vectors > n %
No
Yes
Pan change occured
Camera stationary

Fig. 4. Flowchart of pan change detection algorithm

B. Pan Change Detection

The orientation of each optical flow vector is calculated as $\tan^{-1}\left(\frac{y_i^{t+1} - y_i^t}{x_i^{t+1} - x_i^t}\right)$. The orientations of the optical flow vectors will in general span the entire range of 360^o. The method divides the range of 360^o into eight bins, each representing a sector of width 45^o. A histogram representing the number of flow vectors that fall within each of these eight sectors is computed and is defined as *orientation histogram*. The dominant orientation is obtained by finding the peak value in the orientation histogram.

The pan change of a camera will result in optical flow vectors that have the same orientation. However, deviations may be reported from this behavior when there are moving objects in the scene i.e., not all of the optical flow vectors point in the direction. The number of optical flow vectors that correspond to the dominant orientation should contribute to more than n% of the total number of vectors, if the motion was induced due to pan change. Any motion detected due to moving objects would generally be random i.e., the vectors have random orientations and hence the orientation histogram does not have a predominant peak value. Fig.1(c) and Fig.3(c) show the orientation histograms for the optical flow vectors in Fig.1(b) and Fig.3(b) respectively. The orientation histogram

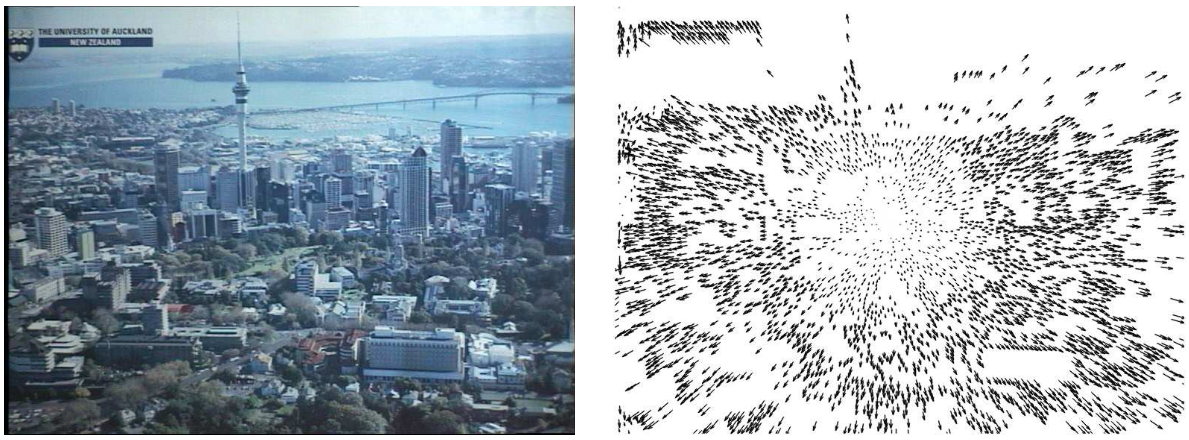

(a) Frame of an image sequence [9] (b) Optical flow vectors

Fig. 5. Divergence of optical flow vectors in the presence of zoom-in

in Fig.3(c) has only one entry as there are no moving objects in the scene while that in Fig.1(c) contains multiple entries corresponding to random movement of waves. From these figures, it is evident that the assumptions made are valid. The flow chart of the pan change detection algorithm is given in Fig.4.

C. Zoom Change Detection

The optical flow field induced due to change of zoom of the camera will deviate from that due to random movement of objects and camera pan change. Zoom-in of the camera lens will result in a divergent field (Fig. 5) while a zoom-out will produce a convergent field. The zoom change has to be first detected and then classified. It should be noted here that convergence and divergence of the flow vectors would be towards and away from the center of a frame respectively. This property is utilized effectively to distinguish convergent and divergent fields.

Let w and h denote the width and height of the camera frame respectively. Then the x- and y- coordinates of the center of the frame (x_c, y_c) are calculated as $x_c = \frac{w}{2}$ and $y_c = \frac{h}{2}$. In the case of a convergent (divergent) vector, the feature point (x_i^{t+1}, y_i^{t+1}) in the frame at time $t+1$ would be nearer (farther) to (x_c, y_c) than the feature point (x_i^t, y_i^t) in the frame at time t. This condition can be represented in mathematical terms as

$$\left|\sqrt{(x_i^{t+1}-x_c)^2+(y_i^{t+1}-y_c)^2}\right| \le \left|\sqrt{(x_i^{t}-x_c)^2+(y_i^{t}-y_c)^2}\right| \tag{9}$$

if Euclidean distance is used. Each flow vector is tagged as converging or diverging depending on this condition. Then the dominant nature of the flow field (i.e., either convergent or divergent) is identified based on the number of vectors that is converging and diverging. If the number of vectors corresponding to the dominant field exceed k% of the total number of vectors, then the camera zoom is detected to have been changed. Fig.6 explains the zoom change detection algorithm.

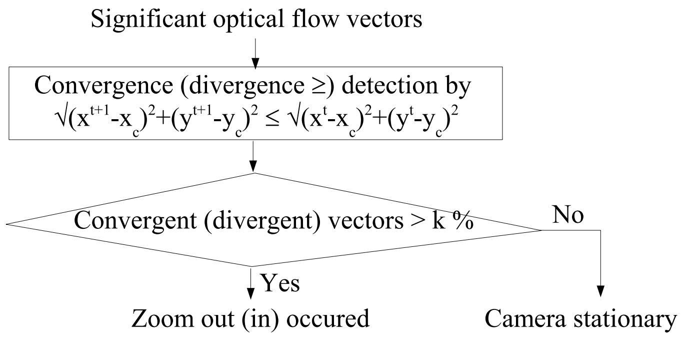

Significant optical flow vectors
Convergence (divergence ≥) detection by
√(x^{t+1}-x_c)^2+(y^{t+1}-y_c)^2 ≤ √(x^t-x_c)^2+(y^t-y_c)^2
Convergent (divergent) vectors > k %
No
Yes
Zoom out (in) occured
Camera stationary

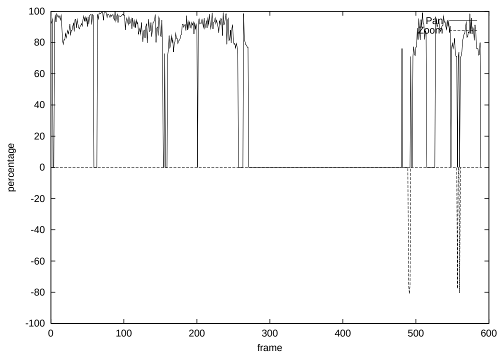
Fig. 6. Flowchart of zoom change detection algorithm

IV. Performance Evaluation

The proposed scheme has been tested extensively using video data collected from different locations. The algorithms are implemented in C++ using OpenCV library [11]. The pyramidal implementation of Lucas and Kanade algorithm (KLT) [12] is used for optical flow estimation. The data sets used in the experiments include diverse indoor and outdoor scenes where significant number of objects move. The events detected by the method are tested manually. The computational performance of the method is evaluated using the time taken per frame. The percentage of flow vectors that correspond to the event being identified are plotted as a function of frame number (Fig.7). The percentages for zoom-out and zoom-in are shown as positive and negative respectively.

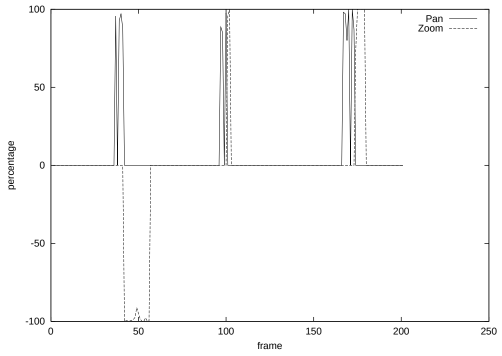
It should be noted here that the flow fields for pan and zoom change are in general complementary, i.e., detection of pan change will avoid the need to check for zoom change and *vice versa*. Hence, pan and zoom change steps need not be performed in parallel but can be run one after the other. However, pan change detection being critical of the two events (it depends on the application again), it is taken to be the first step. The value of ϵ in equation(8) has been set to 1. Sub-pixel ϵ values may be used when the input frame rate of the camera is high.

(a) First frame of data0005 sequence



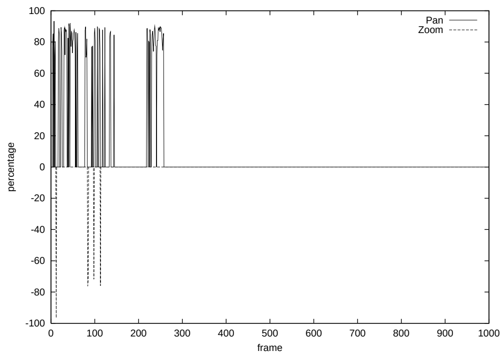
(b) Percentage of flow vectors

(c) First frame of autodome sequence [10]



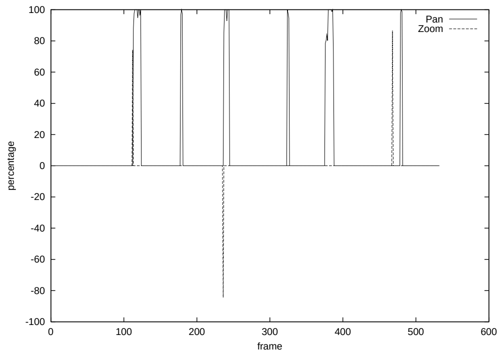
(d) Percentage of flow vectors

(e) First frame of PETS sequence [4]



(f) Percentage of flow vectors

(g) First frame of data1 sequence



(h) Percentage of flow vectors

Fig. 7. Performance of the scheme for various data sets

The data sets used include pan and zoom change events. It is expected that the system does not generate false alarms in the absence of these events. Figures 7(a), 7(c), 7(e), and 7(g) show the first frames of the sequences used. The resolution of these sequences is 8 *bits per pixel* per color component and the dimension is 352×288. The respective percentage plots in Figs. 7(b), 7(d), 7(f), and 7(h) show that the events are detected correctly. It should be noted here that the percentages below the minimum detection percentages m, n, and k (taken as 70%) are set to 0%. Most of the percentages in Fig. 7(d) and Fig. 7(h) are over 90% since the number of moving objects in these indoor scenes is less. Figure 7(f) is an outdoor scene where waves in the sea move; hence the percentages are found to be less.

The computational complexity of the scheme depends on the number of feature points extracted and moving objects in the scene. If the camera is stationary and no moving objects are present in the scene, then the scheme stops after the motion detection step (section III-A). However, if motion is detected, then the pan and/or zoom change detection steps have to be performed and hence the complexity will increase. The complexity of KLT feature point extraction and tracking method increases with the number of feature points. The complexity is more when both the pan and zoom change detection steps are performed for a frame (when motion has been detected but pan was found to be constant).

TABLE I
COMPUTATIONAL COMPLEXITY

Data set	No. of Frames	Average time per frame (secs)
data0005	591	0.157159
autodome	202	0.335089
PETS	1001	0.293973
data1	533	0.130749

Table I shows the complexity of the method for various data sets used. These results were reported on a Dell Optiplex GX 240 computer with Intel Pentium 4 processor (1.6 GHZ) and 512 MB RAM. The results suggest that each frame can be processed in a fraction of a second. The time taken per frame is high for autodome and PETS sequences since motion is detected because of the moving objects but the pan change detection step alone was not conclusive for most of the frames; hence zoom change detection step was also performed. It is less for data0005 and data1 sequences as the high input frame rate permitted the method to stop after motion detection step and the zoom change detection step was not required for most of the frames. The complexity of the scheme can be reduced by substituting the computationally intensive KLT algorithm with other low complex methods.

The scheme assumes that good number of feature points is available in each frame of the video sequence and may not perform well otherwise. The assumption that movement of natural objects such as waves and trees is random holds in most of the cases. The movement of humans can at times be well-directed e.g., a military parade in which case the method may generate a false alarm. Further, if the feature points are available only in a certain portion of the scene, then the method may also raise a false alarm. However, these situations are not very common and the method is expected to perform well in all the remaining cases.

V. CONCLUSIONS

Detection of camera motion is essential for automated analysis of video sequences. Two methods based on optical flow are proposed to detect change in pan and zoom of a camera. The assumption that significant number of flow vectors point in the same direction (orientation) has been used to identify pan change. The convergence and divergence of flow vectors has been utilized for zoom change detection. These assumptions have been verified experimentally using many data sets. Further, the number of false alarms has been found to be minimal. The complexity of the scheme is also shown to be less.

REFERENCES

[1] S.Harasse, L.Bonnaud, A.Caplier, and M.Desvignes, “Automated camera dysfunctions detection,” in *Proc. IEEE Southwest Symposium on Image Analysis and Interpretation*, vol. 1, Lake Tahoe, Nevada, 2004, pp. 36 – 40.

[2] R. Wang and T. Huang, “Fast camera motion analysis in mpeg domain,” in *Proc. International Conference on Image Processing*, vol. 3, Kobe, Japan, 1999, pp. 691 – 694.

[3] V. Kobla, D. Doermann, and A. Rosenfeld, “Compressed domain video segmentation,” Center for Automation Research, University of Maryland, College Park, Tech. Rep. CS-TR-3688, 1996.

[4] T. Boult, “Coastal surveillance datasets, vision and security lab, u. colorodo at colorado springs,” 2005, www.vast.uccs.edu/∼tboult/PETS2005.

[5] S. S. Beauchemin and J. L. Barron, “The computation of optical flow,” *ACM Computing Surveys*, vol. 27, no. 3, pp. 433 – 466, September 1995.

[6] C. Tomasi and T. Kanade, “Shape and motion from image streams: a factorization method - part 3 detection and tracking of point features,” Computer Science Department, Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-CS-91-132, April 1991.

[7] J. Shi and C. Tomasi, “Good features to track,” in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Seattle, WA, June 1994, pp. 593 – 600.

[8] J. Barron, D. Fleet, and S. Beauchemin, “Performance of optical flow techniques,” *International Journal of Computer Vision*, vol. 12, no. 1, pp. 43 – 77, 1994.

[9] “Optical flow image sequences,” ftp://ftp.csd.uwo.ca/pub/vision.

[10] “Autodome autotrack mpeg video,” http://www.autodome.com/DemoVideos/Autotrack.mpg.

[11] “Intel open source computer vision library,” http://www.intel.com/research/mrl/research/opencv/.

[12] J.-Y. Bouguet, “Pyramidal implementation of the lucas kanade feature tracker description of the algorithm,” Microprocessor Research Labs, Intel Corporation, Santa Clara, CA, Tech. Rep., 2000.