An Interactive Framework for Abandoned and Removed Object Detection in Video

Abhineet Kumar Singh and Anupam Agrawal
Indian Institute of Information Technology Allahabad, India
Email: {abhineet.iiita, anupam69}@gmail.com

***Abstract*- This paper describes an interactive application that employs a modular approach to solve the problem of detecting abandoned and removed objects in a video stream in real time. The system breaks down this difficult task into a series of simpler sub-tasks and solves each through a separate module. Since different modules are independent of each other, each may utilize several different methods for solving the corresponding sub-problem. Accordingly, several existing methods have been implemented for each module with the possibility of adding new methods with minimum system reprogramming. Additionally, by switching between these methods in real time, the user can observe and compare the performance and accuracy of different methods and find the optimal combination of methods for a particular scenario. Lastly, the user can also interact with the system to provide useful feedback and help improve its performance when it is not up to the mark. This enables the system to take advantage of the far superior human vision processing capabilities in very complex scenarios while still being completely automated most of the time.**

Keywords—abandoned object detection, video surveillance, modular, interactive

I. INTRODUCTION

Abandoned object detection, from here on referred to as AOD, is one of the most practically useful areas in computer vision due to its application in automated video surveillance systems for the detection of suspicious activities that might endanger public safety, especially in crowded places like airports, railway stations, shopping malls, movie theatres and the like. An abandoned object is usually defined as one that has been lying stationary at a certain place with no apparent human attendance for an extended period of time. Detection of abandoned objects is of prime importance in uncovering and forestalling terrorist activities since it is a reasonable supposition that an abandoned object, if left behind on purpose, may be hiding dangerous items like explosives.

Most existing AOD techniques employ a modular approach with several independent steps where the output of each step serves as the input for the next one. Many efficient algorithms exist for carrying out each of these steps and any single complete AOD system has to address the problem of finding a suitable combination of algorithms to suit a specific scenario. Following is a brief description of these steps and related methods, in the order they are carried out:

a) Background Modeling and Subtraction (BGS): This stage creates a dynamic model of the scene background and subtracts it from each incoming frame to detect the current foreground regions. The output of this stage is usually a mask depicting pixels in the current frame that do not match the current background model. Some popular background modeling techniques include adaptive medians [1], running averages [2], mixture of Gaussians [3, 4], kernel density estimators [5, 6], Eigen-backgrounds [7] and mean-shift based estimation [8]. There also exist methods that employ dual backgrounds [9] or dual foregrounds [10] for this purpose.

b) Foreground Analysis: This method detects false or uninteresting foreground regions produced due to sudden lighting changes and shadows. Several methods exist for detecting sudden lighting changes, ranging from simple gradient and texture based approaches [11, 12] to those that utilize complex lighting invariant features combined with binary classifiers like support vector machines [13]. Shadow detection is usually carried out by performing a pixel-by-pixel comparison between the current frame and the background image to evaluate some measure of similarity between them. These measures include normalized cross correlation [11, 14], edge-width information and illumination ratio [15]. There are many other shadow detection methods as enumerated in [16].

c) Blob Extraction: This stage applies a connected component algorithm to the foreground mask to detect the foreground blobs while optionally discarding too small blobs created due to noise. Most existing methods use an efficient linear time algorithm that was developed in [17]. The popularity of this method is owing to the fact that it requires only a single pass over the image to identify and label all the connected components therein, as opposed to most other methods that require two passes [18, 19, 20].

d) Blob Tracking: This is often the most critical step in the AOD process and is concerned with finding a correspondence between the current foreground blobs and the existing tracked blobs from the previous frame (if any). Many methods exist for carrying out this task, including finite state machines [13], color histogram ratios [21], Markov chain Monte Carlo model [22, 23], Bayesian inference [24, 25], Hidden Markov models [26] and Kalman filters [27].

e) Abandonment Analysis: This step classifies a static blobs detected by the tracking step as abandoned or removed object or even a very still person. The task of distinguishing between removed and abandoned objects is generally carried out by calculating the degree of agreement between the current and background images around the object's edges. There exist several ways to calculate this degree of agreement; two of the popular methods are based on edge energy [28, 11] and region growing [29]. There also exist methods [13] that use human tracking to look for the object's owner and evaluate the owner's activities around the dropping point to decide whether the object is abandoned or removed.

978-1-4799-2275-8/13/$31.00 ©2013 IEEE

II. DESCRIPTION OF THE SYSTEM

This system considers the overall problem of AOD as a collection of smaller problems, each of which can be solved independently of the others. Thus it is built using a different subsystem for solving each of these sub-problems. Similar to the formulation stated in section I, this system can be divided into five main modules along with two additional modules for pre processing and for filtering the final results. A brief description of the algorithms that have been implemented for each of them follows.

1) Pre-processing: This module performs the following two functions:

*a) Contrast enhancement***:** This step helps to improve the quality of low light videos like those taken at night by normalizing the difference between maximum and minimum intensities in the image which in turn helps to increase visibility in darker areas of the scene. It can be carried out using several methods including histogram normalization, image filtering and contrast stretching.

b) Noise Reduction: This step reduces the white noise present in an input frame by smoothing the frame. It is particularly useful for low quality or grainy videos and is also needed to control the amount of noise that becomes visible in low light videos after applying contrast enhancement.

2) Background Modeling and Subtraction (BGS): The system currently includes three different BGS algorithms which are modified to perform object level background updating rather than the typical pixel level one. This is accomplished through a mask of currently tracked objects that is fed back from the blob tracking module (refer section 5.5). A brief description of the three algorithms is presented below:

a) Gaussian Mixture Model (GMM): This method, first introduced in [3] and improved significantly in [4], models the distribution of pixel intensity values over time as a weighted sum of three Gaussian distributions. For each pixel in the current frame, the probability of observing the current intensity is given by:

$$P(X_t) = \sum_{i=1}^{K} \omega_{i,t} * \eta\left(X_t, \mu_{i,t}, \Sigma_{i,t}\right) \quad (1)$$

Here, K is the no. of distributions; $\omega_{i,t}$ is the weight associated with the i^{th} distribution at time t while $\mu_{i,t}$ is the mean and $\Sigma_{i,t}$ is the co-variance matrix of this distribution, η is the exponential Gaussian probability density function given by:

$$\eta(X_t, \mu_t, \Sigma_t) = \frac{1}{(2\pi)^{\frac{n}{2}} |\Sigma_t|^{\frac{1}{2}}} e^{-\frac{1}{2}(X_t - \mu_t)^T \Sigma_t^{-1} (X_t - \mu_t)} \quad (2)$$

Here, n is the dimensionality of each pixel's intensity value (e.g. n=1 for grayscale image and n=3 for RGB image). Further details can be found in [3]. In the faster version of this method, presented in [4], the distributions are ordered simply by their weights, rather than ω/σ used in [3], thus simplifying the sorting procedure without any significant impact on performance.

b) Detecting Shadows: Two different shadow detection methods have been implemented in this system, both utilizing the normalized cross-correlation (NCC) of intensity values between the grayscale versions of the current frame and the background image. The first and more complex one is described in [11] while the other simpler one in [14].

b) Adaptive Median: This BGS method, described in [1], works under the assumption that the background is more likely to appear at any given pixel over a period of time than foreground objects, i.e. the past history of pixel intensity values is likely to contain maximum occurrences of the background intensity at the pixel location. This leads to the reasonable supposition that the pixel stays in the background for more than half the values in its history. The median of previous n frames can therefore be used as the background model.

Due to the high memory requirements of the iterative version, a recursive version of this algorithm is more practically feasible. Following is the update equation for the background model in this approach:

$$B_t = \begin{cases} (B_{t-1} - 1)\ if\ B_{t-1} > I_t \\ B_{t-1}\ if\ B_{t-1} = I_t \\ (B_{t-1} + 1) if\ B_{t-1} < I_t \end{cases} \quad (3)$$

Here, B_t and I_t respectively refer to the intensity values in background model and the current frame at time t.

c) Running Gaussian Average: The basic idea here is to use the average of the last n frames as the background model. Since recent frames are more likely to contribute to the current background than older ones, a weighted average is used with higher weights attached to more recent frames. When these weights vary according to the Gaussian distribution, the running Gaussian average is obtained. The algorithm implemented in this system is detailed in [2]. The background model here is updated according to the following equations:

$$\mu_{t+1} = \alpha I_t + (1 - \alpha)\mu_t \quad (4)$$

$$\sigma^2{}_{t+1} = \alpha(I_t - \mu_t)^2 + (1 - \alpha)\sigma^2{}_t \quad (5)$$

Here, μ_t and $\sigma^2{}_t$ respectively refer to the mean and variance of the Single Gaussian distribution while α is the learning rate

3) Foreground Analysis: The noisy portions of the BGS output may contain false foregrounds produced for example during sudden lighting changes (a light is switched on or off) as well as actual foregrounds (like shadows) that are of no interest to further processing but can complicate it significantly. Thus, a separate foreground analysis stage is needed to remove such false and uninteresting foreground pixels from it. This process is further divided into the following three distinct tasks:

a) Detecting sudden lighting changes: When a new light source is suddenly introduced into a scene, it creates a patch of light that is detected as a foreground region since its color intensity values are significantly different from those of the existing background model. However, though the actual RGB values may have changed, the underlying texture of the region remains unchanged and this fact can be utilized to detect such regions. A texture difference measure, proposed in [11], is used for this purpose.

c) Morphological frame processing: Both lighting change detection and shadow detection processes are prone to false classifications and often leave behind 'holes' inside valid foreground objects along with some left behind shadow pixels that may be detected as small blobs. These are removed by applying the morphological operations of closing followed by opening.

4) Blob Extraction: This stage detects connected components in the foreground mask to extract meaningful objects while discarding any blobs that are smaller than a specified threshold. A simple, efficient but fairly accurate algorithm described in [17] is used for this purpose.

*5) Blob Tracking***:** This stage tracks only the static objects in the scene using a heavily modified version of the algorithm used in [9]. Following are the main steps in this process:

a) Establish blob correspondence: The first step is to compare each blob in the incoming frame with the existing blobs in the tracking system to find a match based on position and size. For an existing blob to match a new blob, their areas and positions must differ by less than a threshold. An additional constraint is imposed on the matching process that each existing blob may match at most one new blob and vice versa

b) Update state variables for existing blobs: Three state variables are maintained for tacked blobs hit count, miss count and occluded count. When a match is found for an existing blob, its hit count is incremented by one while its miss and occluded counts are set to zero. All new blobs that do not match any existing blob are added to the tracking system with their hit counts initialized to one.

An existing blob that does not match any of the new blobs is considered to be occluded if more than a certain fraction (say 0.8) of the blob's pixels has been detected as foreground in the current frame. In this case, both its hit count and occluded count are incremented by one. If the object is not detected as occluded, its miss count is incremented by one.

c) Remove non-static objects: An object is discarded from the tracking system if one of the following conditions is satisfied:

1. Not detected/matched for several consecutive frames: This occurs when its miss count exceeds either a user specified threshold or the current hit count.

2. Occluded for too long: This occurs when the occluded count exceeds a threshold.

3. Appearance changes significantly: The change in a blob's appearance is measured by the mean difference in pixel intensity between the stored image and the current frame averaged over all the pixels in the blob's bounding box. If this difference exceeds a threshold, the object is immediately discarded. An exponential moving average of these mean differences is also maintained for use by the abandonment analysis module for distinguishing between actual static objects and very still persons.

d) Change object label: If the hit count of an object exceeds a user defined threshold, it is labeled as static and its maximum permissible occluded and miss counts are multiplied by respective constant factors to make greater allowance for accidental misses and prolonged occlusions. If it exceeds a second higher threshold, this object is passed to the abandonment analysis module to be classified as abandoned or removed.

e) Provide feedback to the BGS module: To prevent static objects from being learnt into the background, all the tracked objects that are detected in the current frame fed back to the BGS module so that the background model is not updated for pixels contained in their bounding boxes. An object as a whole is pushed into the background if it is detected as a state change or if its alarm timeout has been reached. This feedback enables the BGS algorithm to perform object-level (as opposed to pixel-level) background updating and thus helps to avoid foreground fragmentation.

*6) Abandonment analysis***:** This is the final stage whose purpose is to prevent false detections of abandoned objects due to the 'ghost effect' created when an object in the background is removed, leaving behind a false foreground object. It also checks the blob's internal variation during the period it has been tracked to ensure that it is not a very still person. This system therefore can be divided into two subsystems:

a) Removed object detection: When a background object is removed from the scene, it is reasonable to assume that the area thus vacated will now exhibit a higher degree of agreement with its immediate surroundings than before. This assumption can be used to classify an object as abandoned or removed by evaluating, in both the background image and the current frame, a measure of agreement of the object's edge (and nearby interior) pixels with their immediate neighborhood pixels (outside the object). Following two methods have been implemented to measure this degree of agreement in this system:

i) Region Growing: This method was introduced in [29] and consists of a two step process. In the first step, the blob's boundary is eroded to obtain a set of pixels that lie near the boundary but are completely inside the object. These pixels are then used as seed points in a standard region growing algorithm to add all the surrounding pixels that are similar to these. The object is considered removed if the region growth is significantly more in the current frame and abandoned otherwise.

ii) Edge Detection: This method, used in [11, 28] ,works on the idea that placing an object in front of the background will introduce more edges to the scene around the object's boundaries, provided that the background is not extremely cluttered. Two methods have been implemented for edge detection: one is a simple gradient based method that uses Sobel operator to obtain separate x and y gradient images while the other one uses Canny edge detection [30].

b) Still Person Detection: An exponential running average of mean pixel differences between the current frame and the stored image is maintained for each object in the tracking system. The basic idea here, introduced in [31] is that while the appearance of a true static object remains completely unchanged while a false static object detected due to a very still

person will show some variations in the pixel values within the constant bounding box. These internal blob variations are captured by the running average of mean differences and if this value exceeds a certain threshold, the object is classified as a still person.

7) Blob Filtering: The abandonment module classifies a static blob into one of four categories: abandoned, removed, state change or still person. A blob detected as a state change or a still person is removed from the tracking system. If a state change, it is also pushed into the background. An object detected as abandoned causes an alarm to be raised immediately while one detected as removed is first passed through a filtering process where it is compared to each of the user-specified regions in the scene (provided that any have been added to the filtering module). An alarm is raised only if it matches one of them. This matching can be done on the basis of the blob's position, size, appearance or any combination of these, as specified by the user.

III. TESTING AND RESULTS

A. Setup

Since the system features multiple methods for several of its modules, it is not possible to provide the results of every possible combination of these methods. Therefore, some preliminary testing was done and the following combination of methods was found to give good results in most cases:

- Pre-processing: Both contrast enhancement and noise reduction were turned off except in a few night/low light scenes.
- BGS: modified GMM introduced in [4] (sub-section 2.1 in section II (A)).
- Shadow detection: Simple NCC based method (sub-section 3.2 in section II (A)).
- Morphological processing: Both closing and opening operations were enabled (sub-section 3.3 in section II (A)).
- Abandonment analysis: Canny edge detection (sub-section 6.1.2 in section II (A)).

The results presented here have been obtained using this configuration for all the tests with only a few minor modifications for specific cases.

B. Datasets

This system has been tested on 4 different datasets, 3 of which are publicly available benchmark datasets while one is a custom prepared dataset with videos of various indoor and outdoor scenarios featuring both day and night time scenes. Following is a brief description of these datasets:

1) PETS 2006: This dataset, available at [32], contains videos of 7 different events each captured from 4 different viewpoints. The videos depict various left-luggage scenarios of increasing complexity captured in a train station. A wide range of items including briefcase, suitcase, rucksack and even a ski gear carrier constitute the left-luggage in these videos.

2) PETS 2007: This dataset, downloadable from [33], contains videos of four scenarios: loitering, attended luggage removal, luggage exchange, and unattended (or abandoned) luggage. There are two events for each of these scenarios with each event having been captured from 4 angles. Only the last two sets of videos (sets 7 and 8) depict abandoned object events and hence only these have been used for testing.

3) AVSS i-LIDS: This dataset is available at [34] and consists of CCTV footage of two scenarios: abandoned baggage (AB) and illegally parked vehicles (PV). There are 3 different videos of increasing difficulty levels for both of these scenarios with the latter also having a night time video. Testing was done on both scenarios since a parked vehicle can also be considered as an abandoned object.

4) Custom dataset: This dataset contains 18 videos out of which 12 are outdoor scenes while 6 are indoor ones; 13 of these were shot during in well lit conditions while 5 were shot at night or in poorly lit interiors. Most videos feature both abandoned and removed object events.

Videos in the first three datasets have original resolution of 720x576 but have been resized to 360x288 for testing. The custom dataset videos were shot at 640x480 and then resized to 320x240 for testing.

C. Results

1) Summary: Experiments were carried out on an Intel Core i5 processor clocked at 2.5 GHz and having 4 GB of RAM. Processing was carried out at both the original resolution (360x288 or 320x240) as well as at half the resolution in each dimension (180x144 or 160x120). The results have been summarized below in Tables 2 and 3.

Set		Actual events	True detections	False detections	
				Still Person	Other objects
PETS2006					
1		4	4	0	1
2		4	4	0	0
3		0	0	0	0
4		0	0	1	0
5		4	4	1	1
6		4	4	0	1
7		4	3	0	1
Total		**20**	**19**	**2**	**4**
PETS2007					
7		4	3	0	0
8		4	3	0	0
Total		**8**	**6**	**0**	**0**
AVSS i-LIDS					
AB	Easy	1	1	0	0
	Medium	1	1	0	0
	Hard	1	1	1	1
	Total	3	3	1	0
PV	Easy	1	1	0	0
	Medium	1	0	0	1
	Hard	1	0	0	1
	Night	1	1	0	0
	Total	4	2	0	2
Total		**7**	**5**	**1**	**3**
Custom dataset					
1		1	1	0	0
2		1	1	0	0
3		0	0	0	0
4		2	2	0	0
5		2	2	0	0
6		2	1	0	0

7	2	0	0	0
8	2	2	0	0
9	1	0	0	1
10	2	2	0	0
11	2	2	0	0
12	2	2	0	0
13	2	2	0	0
14	2	2	1	0
15	2	0	0	0
16	8	8	0	1
17	1	1	0	0
18	1	0	0	0
Total	**35**	**28**	**1**	**2**
Overall	**70**	**58**	**4**	**9**

Table 2 Result summary for all the datasets

Category	**Actual events**	**True detections**	**False detections**	
			Still Person	**Other objects**
Outdoor	19	15	0	1
Indoor	16	13	1	1
Daytime / Well lit	26	22	1	2
Night/Dark	9	6	0	0

Table 3 Category wise result summary for custom dataset.

. *2) Analysis*: A modified version of the quantitative metric introduced in [9] has been used here to measure the overall performance of this system. This metric evaluates the number of successfully detected events, penalized by the false detections, as a fraction of the total number of events. It is given by:

$$S = \frac{true\ detections - (0.5 * n_p + 0.25 * n_b)}{total\ no.of\ events} \quad (6)$$

Here, $n_p$and n_b respectively represent the no. of still persons and other objects falsely detected as abandoned or removed objects. The performance figures obtained for each of the datasets are listed in Table 4. As can be seen there, the performance was worst in the AVSS-PV dataset. The main reason for this was that the camera that recorded these videos was moving or shifting slightly every now and then, probably due to winds, thus leading to small changes in the viewpoint. Since this system tracks only static objects based on their precise position in the scene, such shifts caused the system to lose track of existing static objects and start tracking them all over again.

Dataset		**System performance (S) in percent**
PETS2006		85.00
PETS2007		75.00
AVSS i-LiDS	AB	83.33
	PV	37.50
	Overall	45.83
Custom	Outdoor	77.63
	Indoor	76.56
	Well Lit	80.76
	Dark	66.67
	Overall	72.72
All datasets combined		**76.79**

Table 4 Performance summary for different datasets.

3) Comparison with existing methods: Though both PETS2006 and AVSS i-LiDS datasets have been used for testing in many contemporary works in literature, very few of these report sufficiently detailed results for direct comparison with our results. For example, even though PETS2006 has been used in [9], [10], [21], [22], [24] and [29], the results for all 4 views have not been reported in even one In addition, some, like [24] and [21], have provided no information about false positives.

Combined results for PETS2006 and AVSS-AB datasets have been reported in [9]. Using a similar but slightly more relaxed metric, the overall accuracy reported there is 85.20%. The corresponding figure for our system is 84.78%.though it must be noted that this has been obtained using a stricter metric. The results in [10] have been reported for AVSS-AB dataset with accuracy of 66.67% which is significantly worse than our result of 83.33%. The results of AVSS-PV and PETS2006 datasets have also been reported here but only partially and cannot be compared with our results. The system presented in [21] has been tested on PETS2006 and PETS2007 datasets but no information has been provided about false positives or individual views, again rendering it unsuitable for comparison with our system.

Results in [22], [24] and [29] have been reported for only one view for each of the 7 scenarios in PETS2006 dataset. In addition, the actual false positive count has not been reported in [22] and [24]. The approximate accuracy figures are 80% for [22] and 91.67% for [24] and 90% for [29]. These are comparable to our accuracy of 85% obtained over all the 4 views; choosing only the best result for each scenario would increase it to 100%. The system in [29] has also been tested on AVSS dataset with accuracy of 58.93% which is only slightly better than our result of 57.14% in spite our system's failure to cope with the shaky nature of AVSS-PV dataset videos.

D. Conclusion and future scope

The application was tested on a large number of publicly available as well as custom made videos and was found to give accuracy comparable to most contemporary systems, while still managing to run in real time on a fairly modest setup. One particularly noteworthy aspect of its performance was the very low rate of false positives that was obtained. Also, these results were obtained by using virtually the same methods and parameter values across all the videos, with only minor changes for some cases. Significantly better results can be obtained if exhaustive testing and adjustments are carried out for each scenario.

Finally, and perhaps most importantly, the methods that have been presented and tested in this paper are those that are part of the initial implementation of this system. These were specifically chosen to be relatively simple methods, both to implement and to execute, due to limitations of time and computational resources. However, owing to the modular nature of this system, it is quite easy to add more sophisticated methods to any of its module. This system can, therefore, be considered as the foundation for a truly robust framework that only requires a bit of calibration to perform well in practically any scenario.

REFERENCES

[1] N. McFarlane and C. Schofield, "Segmentation and tracking of piglets in images," *Machine Vision and Applications* Vol. 8, Issue 3, pp. 187–193, 1995.

[2] C.Wren, A. Azarbayejani,T. Darrell, and A. Pentland, "Pfinder: real-time tracking of the human body," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 19 Issue 7, pp. 780-785, July 1997.

[3] C. Stauffer and W. E. L. Grimson, "Adaptive background mixture models for real-time tracking," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, Vol. 2, pp. 246–252, Feb. 1999.

[4] Z. Zivkovic, "Improved adaptive Gausian mixture model for background subtraction," in *Proc. International Conference on Pattern Recognition*, Vol. 2, pp. 28–31, Aug. 2004.

[5] A. Elgammal., D. Harwood and L.S. Davis, "Nonparametric background model for background subtraction," in *Proc. 6th European Conference on Computer Vision*, Vol. 2, pp. 751-767, 2000

[6] A. Elgammal, D. Harwood, R. Duraiswami and L.S. Davis, "Background and Foreground Modeling Using Nonparametric Kernel Density Estimation for Visual Surveillance," in *Proc. The IEEE,* Vol. 90, Issue 7, pp.1151-1163, Jul. 2002.

[7] N. M. Oliver, B. Rosario, and A. P. Pentland, "A Bayesian Computer Vision System for Modeling Human Interactions," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 22, Issue. 8, pp. 831-843, Aug. 2000

[8] B. Han, D. Comaniciu, and L.S. Davis, "Sequential kernel density approximation through mode propagation: applications to background modeling," in *Proc. Asian Conference on Computer Vision*, Jan. 2004.

[9] A.Singh, S.Sawan, M.Hanmandlu, V.K.Madasu, B.C.Lovell "An Abandoned Object Detection System Based On Dual Background Segmentation," in *Proc. Sixth IEEE International Conference on Advanced Video and Signal Based Surveillance*, pp. 352-357, 2009.

[10] Fatih Porikli, Yuri Ivanov and Tetsuji Haga "Robust Abandoned Object Detection Using Dual Foregrounds," *EURASIP Journal on Advances in Signal Processing*, 2008.

[11] Y. Tian, M. Lu and A. Hapapur, "Robust and efficient foreground analysis for real-time video surveillance," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, vol. 1, pp. 1182-1187, June 2005.

[12] L. Li and M.K.H. Leung, "Integrating intensity and texture differences for robust change detection," *IEEE Transactions on Image Processing*, Vol. 11, Issue 2, pp. 105-112, Aug. 2002.

[13] Q. Fan and S. Pankanti, "Robust Foreground and Abandonment Analysis for Large-Scale Abandoned Object Detection in Complex Surveillance Videos," in *Proc. Ninth IEEE International Conference on Advanced Video and Signal-Based Surveillance*, pp. 58-63, Sept. 2012.

[14] J.C.S. Jacques, C.R. Jung and S.R. Musse, "Background Subtraction and Shadow Detection in Grayscale Video Sequences," in *Proc 18th Brazilian Symposium on Computer Graphics and Image Processing*, pp. 189-196, Oct. 2005.

[15] J. Stander, R. Mech and J. Ostermann, "Detection of moving cast shadows for object segmentation," *IEEE Transactions on Multimedia*, Vol. 1, Issue 1, pp. 65-76, Mar. 1999.

[16] N. Al-Najdawi, H. E. Bez, J. Singhai and E. A. Edirisinghe "A survey of cast shadow detection algorithms," *Pattern Recognition Letters*, Vol. 33, Issue 6, pp. 752-764, April 2012.

[17] F. Chang, C. Chen and C. Lu, "A linear-time component labeling algorithm using contour tracing technique," *Computer Vision and Image Understanding,* Vol. 93, Issue 2, pp. 206-220, Feb. 2004.

[18] A. Rosenfeld and P. Pfaltz, "Sequential operations in digital picture processing," *Journal of the Association for Computing Machinery*, Vol. 13, Issue 4, pp. 471-494, Oct. 1966.

[19] C. Fiorio and J. Gustedt, "Two linear time Union-Find strategies for image processing," *Theoretical Computer Science*, Vol. 152, Issue 2, pp. 165-181, Feb. 1996.

[20] R. Lumia, L. Shapiro and O. Zuniga, "A new connected components algorithm for virtual memory computers," *Computer Vision, Graphics, and Image Processing*, Vol. 22, Issue 2, pp. 287-300, May 1983.

[21] Xuli Li, Chao Zhang, Duo Zhang "Abandoned Objects Detection Using Double Illumination Invariant Foreground Masks," in *Proc. 20th International Conference on Pattern Recognition*, pp. 436-439, Aug. 2010.

[22] Smith, K., Quelhas, P. and Gatica-Perez, D, "Detecting Abandoned Luggage Items in a Public Space," in *Proc. Ninth IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pp. 75 – 82, June 2006.

[23] Z. Khan, T. Balch and F. Dellaert, "MCMC-based particle filtering for tracking a variable number of interacting targets," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 27, Issue 11, pp. 1805-1819, Nov. 2005.

[24] F. Lv, X. Song, X., B. Wu, V. Kumar and R. Singh, "Left Luggage Detection using Bayesian Inference," in *Proc 9th IEEE International Workshop on Performance Evaluation of Tracking and Surveillance*, pp. 83—90, June 2006.

[25] R. Singh, S. Vishwakarma, A. Agrawal and M. D. Tiwari, "Unusual activity detection for video surveillance," in *Proc. 1st International Conference on Intelligent Interactive Technologies and Multimedia*, pp. 297-305, Dec. 2010.

[26] D. W. Park, J. Kwon and K. M. Lee, "Robust visual tracking using autoregressive hidden Markov Model," in *Proc. IEEE Conference on Computer Vision and Pattern Recognition*, pp. 1964-1971, June 2012.

[27] Z. Han, Q. Ye and J. Jiao, "Online feature evaluation for object tracking using Kalman Filter," in *Proc. 19th International Conference on Pattern Recognition*, pp. 1-4, Dec. 2008.

[28] J. Connell, A.W. Senior, A. Hampapur, Y.-L. Tian, L. Brown and S. Pankanti, "Detection and Tracking in the IBM PeopleVision System," in *Proc. IEEE Conference on Multimedia and Expo*, Vol. 2, pp. 1403-1406, June 2004.

[29] Y. Tian, R. Feris and A. Hampapur "Real-time detection of abandoned and removed objects in complex environments," in *Proc. IEEE International Workshop Visual Surveillance* (in conjunction with ECCV'08), Marseille, France, 2008.

[30] J. Canny, "A Computational Approach to Edge Detection," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, Vol. 8, Issue 6, pp. 679-698, Nov. 1986.

[31] N. Bird, S. Atev, N. Caramelli, R. Martin, O. Masoud and N. Papanikolopoulos, "Real Time, Online Detection of Abandoned Objects in Public Areas," in *Proc. IEEE International Conference on Robotics and Automation,* pp. 3775-3780, May 2006.

[32] "PETS 2006 Benchmark Data," http://www.cvg.rdg.ac.uk/PETS2006/data.html, June 2013.

[33] "PETS 2007 Benchmark Data," http://www.cvg.rdg.ac.uk/PETS2007/data.html, June 2013.

[34] "2007 IEEE International Conference on Advanced Video and Signal based Surveillance (AVSS 2007)," http://www.eecs.qmul.ac.uk/~andrea/avss2007_d.html, June 2013

.