Efficient Method for Counting and Tracking of Moving Objects through Region Segmentation

Ravi Kiran Boggavarapu
School of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
Email: vsravikiran.b@gmail.com

Pushpendra Kumar Pateriya
School of Computer Science and Engineering
Lovely Professional University
Phagwara, Punjab, India
Email: pushpendra.14623@lpu.co.in

***Abstract*—Object Counting and Tracking are an integral part of computer-aided surveillance intelligence. In this paper, we propose an Object Counting algorithm based on region segmentation and a tracking method that works on the Kalman Filter to predict the next state of a moving object. We considered people, moving under an overhead camera as our primary targets. As a pre-processing step, we extracted foreground objects using a background subtraction method and optimized those foreground objects using Morphological operations. The counting algorithm works by segmenting the image into regions and analyzing the individual regions to count the moving objects based on certain heuristics. Experiments show that our method can obtain counting accuracy about 89.47% with a decent performance.**

***Index Terms*—Moving Object Counting, Object Counting, Object Tracking, Region Segmentation, Kalman Filter**

I. Introduction

Object Counting and Tracking is one of the most challenging and key application of computer-aided surveillance intelligence. With the rapid increase in the number of security cameras in both public and private places, there is a necessity to develop a light-weighted yet reliable algorithm for counting and tracking objects. There are two contributions of this paper; the first one is a novel region-based object counting method to count the moving objects, such as pedestrians, in a surveillance video. The second contribution is a tracking method based on the Kalman Filter. The system setup is shown in Fig. 2.

Several methods based on Distributed Wireless Sensor Networks(WSN) are widely used to count the moving objects, ranging from contact sensors to using photocells, and passive infrared waves (PIR). However, most of these sensors provide a limited capability, such as detecting the emptiness or occupancy of an area. Contact sensors can be efficiently used to count the individual moving objects, but they limit the flow of objects. Therefore, multiple contact sensors are required to suffice the situation and deal with the flow limitation, therefore increasing the cost of the system. There are other methods using photocells and PIR; these techniques can be ineffective to handle occluded objects. One way to deal with the problem of occlusion is to use high-level sensors such as laser scanners as utilized in the reference [16]. Although efficient, laser scanners can be costly, and many situations require installation of more than one laser scanner.

978-1-5090-0612-0/16/$31.00 ©2016 IEEE

On the other hand, Computer Vision and Image processing techniques can be helpful while dealing with the cost-effectiveness of the system while producing fair results. Therefore, many counting methods have been proposed so far that are entirely based on Vision and Image Processing techniques:

Multiple studies present different Object counting and tracking techniques using cameras. Bin Li et al., used a method that combines head detection and cross-line judgment to estimate the number of people moving under an overhead camera in a room [4]. Another work used a similar kind of approach as we did, i.e., a region-based counting approach [5] - The authors in [5] followed a simplistic approach by limiting the region of interest to a small area, which, however, might not work well with multiple objects; the method would not be able to count multiple objects moving in the region at same time.

Other applications of object counting based on region segmentation have been proposed. David Schreiber et al. implemented a luggage counter using a CPU rendered background subtraction method to extract the foreground objects [6]. For counting the moving luggage, they have developed an eGate, which makes a cross line judgment to count the moving luggage. Yoginee et al. [7] proposed a region-based segmentation method to count the number of moving cars by segmenting and normalizing the RGB blocks in every frame.

Referenced in [17] proposed a tracking method that uses an appearance vector based on color Region of Interest(ROI) and a probabilistic model to get the modeling-appearance and spatio-temporal aspects of the image. The authors of [17] agreed that their results could be less confident in the presence of occlusions and ambiguities. Another work proposed in [18] used multiple cameras to avoid occlusion and determines the count of the people through the occupancy volume of a crowded place. With the help of background subtraction, this method extracts the foreground objects. A graph produced by these foreground objects is used to get the estimated count of people in the area. The methods proposed in [17] and [18] uses stereo cameras. Stereoscopic camera networks included more complex installation and also increases the final price of the system.

Further, monocular vision is cost-effective. Many studies [19-22] show that using monocular vision can produce a similar and, in some cases, better [19] results. Referenced in

[20], the authors have used an overhead camera and proposed an object counting method based region segmentation and achieved an outdoor accuracy of 78% with an overall accuracy of84%. Several other works proposed in [22][21] also uses a cross-line judgment to count the moving people. To overcome the problem of occlusion, work proposed in [21] uses a tilted camera and a Support Vector Machine(SVM) to detect multiple heads. The methods in [22][21] lack a shadow detection method, and this could lead to ambiguous results in outdoor conditions. As the author in [21] states that their system has an unresolved problem. Another excellent work proposes in [22] uses The Kalman filter to count the tracked heads. It detects the circular shape of the head and uses a cross-line judgment to count the moving heads. Although this method works well, the accuracy of the count falls to 87% as multiple numbers of people per area increases.

The method proposed in this paper is entirely novel in the sense that we not only used the cross-line judgment, as many successful methods do, but we also took certain heuristics such as the pace of the moving object and blob dimensions of the object. These heuristics(described in Section II) are not only cost(computational cost) efficient but also makes our system more reliable to count multiple objects. It is also worth noting that the methods that require less computational cost can even run on low-powered devices. Further, we achieved an accuracy of 89.47%, which is slightly higher than the other methods mentioned in the above paragraph. Additionally, our method is capable of detecting multiple objects and it also efficiently deals with shadows.

II. Proposed Method

The proposed method consists of several steps and the overview of these steps is represented using a flowchart in Fig 1. In this section, we discuss those steps involved.

A. Background Subtraction

Background subtraction is a primary vision task that works on statistical analysis of pixels, also known as Gaussian Mixture Models(GMM) [2]. This method generates a density function for the pixels that belongs to contiguous frames, and a pixel is considered as background if its value falls under this density function. Any pixel value that does not belong to the density function is considered as a foreground pixel. We used the method proposed in [1] to detect the background pixels. A simple background detection as stated in [2] works on the following formula:

$$B(x,y,t) = \frac{1}{t}\sum_{t'=1}^{t} I(x,y,t') \tag{1}$$

Where $B(x,y,t)$ denotes a Background pixel for an instantaneous pixel $I(x,y,t')$ at a point (x,y) and time t.

Even though this approach works well under certain conditions, change in illumination conditions can cast shadows that can result in false positives. Therefore shadow detection is an integral part of any efficient background subtraction method [9]. Additionally, several other factors impact the accuracy of foreground object detection. Such as selecting of some components, also known as mixtures, a threshold value for shadows.

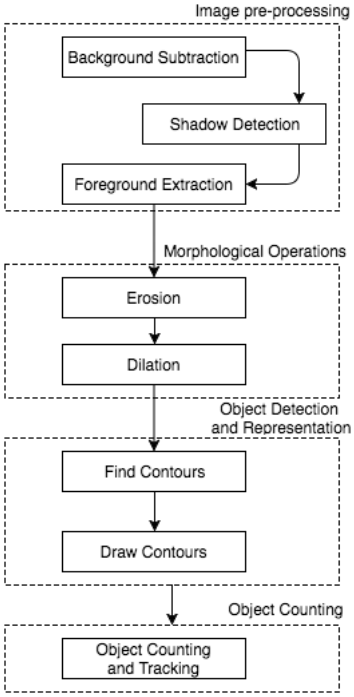

Fig. 1. Flowchart representation of the proposed method.

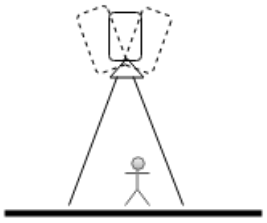

Fig. 2. This figure illustrates the system setup. An overhead camera is mounted on the top in an angle between 70° to 110°

B. Morphological Operations

Background subtraction results in a binary image where non-static objects can be found as binary large objects(blobs). After extracting the foreground parts of the image, as shown in Fig. 10, the frame is subjected to morphological operations; these operations are useful to separate overlapped blobs(erosion), and combine weakly formed or disconnected blobs(dilation).

Morphological operations primarily deal with the application of a structuring element to an image and relative ordering of the pixels, rather than their numeric value. Structuring element, also known as a kernel, is a relatively small binary image, such as a binary matrix [3]. The two basic morphological operations are:

1) Erosion: Erosion is a basic morphological operation used in Digital Image Morphology, which is used to shrink

the image region. A structuring kernel is applied to an image region that results in shrinking of the number of pixels existing at the boundary of the image.

2) Dilation: Unlike erosion, dilation operation enlarges the boundaries of an image region. The area around the foreground pixels of an image region increase in size, while the holes inside this area are reduced in size. The resultant outputs of morphological operations are presented at the end of the paper in Fig. 10 and 11.

C. Object Detection and Representation

Contour detection works on boundary detection algorithm [8], which derives a series of points, also known as chain-codes, from the boundary of the object. The contours are formed as a result of the link between a connected component of 1-pixels and a connected component of 0-pixels. After finding the contours in a binary image, they are represented using the Ramer-Douglas-Peucker algorithm [10] [11] to approximate a curve around the detected contours from the previous points. These approximated polygon points are further used to draw a polygon, which in our case is a rectangle.

D. Object Counting

The proposed algorithm is presented in Fig. 5 and Fig. 6.The object counting method is based on segmentation of the entire frame into sub-parts known as Regions of Interest(ROI), as shown in Fig. 3. ROIs are particularly useful because they help us to focus on a specific segment at a time rather than the entire image. This has many advantages:

- Fast calculations: Since, we are working on an ROI, which is relatively smaller in size as compared to the whole image, the computations are particularly faster. Especially, computations such as finding contours and storing contours can occupy large space in a case when the whole image is considered.
- Less noise: If the image is quite large, there are chances that there is a relatively large amount of evenly distributed noise. However, in our case, using ROI we are ensured that we are working on a relatively small amount of noise.

After finding ROIs, the entire image is segmented into different regions and any objects moving in-between those ROIs are counted. However, a count produced by such a method can be ambiguous and not reliable. Hence, we have included some heuristics in counting the moving objects.
Note: Please refer to the algorithm in Fig. 5 and Fig. 6. from here.

The first heuristic is the area of contour: After representing the contours as mentioned in Section II(c), the area of each moving contour is obtained and stored in $MINCAREA$ in the algorithm shown in Fig. 5. This area is used as an elimination step. Background subtraction requires that the video source, i.e., the camera, in our case, should be fixed(or stable). Since, the background subtraction obtains the time-averaged background, any moving object between consecutive frames is considered as a foreground object.

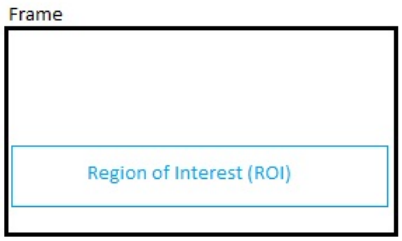

Fig. 3. This is an illustration of how we segmented the entire image in to Regions of Interests.

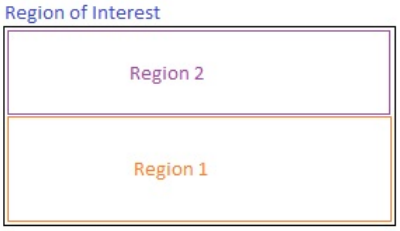

Fig. 4. This is an illustration of how we further segmented the ROIs into two regions.

Therefore, if a video is not stable, it may lead to errors in the counting. To prevent this, we have set a threshold value for the area of contour to consider it as a positive target.

As said above, ROI is further segmented into two regions, named Region1 and Region2 (as shown in Fig. 4). Whenever an object enters the ROI, it contour's center point is calculated and stored in loc; which holds the location of the moving object. $currentState$ is a flag that can contain three value (1, 2, or 3). So, if the value of $currentState$ is 0 and the blob is in Region 1, then $currentState$ is assigned 1 and the current time of the system is stored in $startTime$. This is our second heuristic; we assumed that the objects are moving continuously in the ROI. So, after performing calculations, we concluded that the average time for a moving object to cross the region of interest is less than 2 seconds in a 30fps video.

Next, if the value of $currentState$ is 1 and the blob is in Region 2, the again the current time of the system is taken and is stored in a $stopTime$. If the difference between the $startTime$ and $stopTime$ is more than two, then the flag $currentState$ will be to 0 to prevent a statistical hypothesis errors, else a $UpdateCounter()$ is called upon.

The $UpdateCounter()$ method checks all the detected contours in the ROI for the cases of overlapping - This is our third heuristic; after evaluating samples, we determined the average areas for a single, as well as overlapped multiple blobs. Therefore, $UpdateCounter()$ method checks if any blobs are overlapped, and helps attain multiple-object counting.

E. Object Tracking

To track a moving object, we need an efficient data fusion algorithm for predicting the next state. Although, there are methods like Camshift [12] that estimate the next state of the

moving object, they lack an efficient convergence procedure. Therefore, because of its small computational requirement, elegant recursive properties, and optimal convergence, we chose the Kalman filter [13], which is one of the commonly and widely used estimators [14].

The Kalman Filter algorithm works on Gaussian distribution, where the state space of latent variables is continuous; permitting an exact inference in a Bayesian system. In simple words, The Kalman filter uses a Bayesian system along with the other parameters such as latent variables. In statistics, Latent variables are not observed but inferred through a mathematical model; for example, a latent variable of a moving object can be its speed, acceleration, and steering angle. According to the Kalman filter model, the state of a system at a time t resulted from the previous state at time $t-1$ according to the equation:

$$s_t = T_t s_{t-1} + C_t i_t + n_t \tag{2}$$

Where s_t denotes the current state of the system contained in a state vector, i_t denotes the control input(such as acceleration and steering angle), and n_t contains the process noise for each parameter of the state vector. T_t and C_t are transition matrices, which applies the effect of latent variables(variables in the control input) and previous state(t-1) on the current state.

Kalman filter algorithm involves two states: Prediction and Measurement update; these combine the system models, such as Bayesian, or, Hidden Markov, with noisy measurements of system variables to provide an optimal estimation of the next state. A detailed introduction to Kalman filter is provided by Welch and Bishop [15] and [14]. Discussion of tracking results is presented in Section III(B).

III. Experimental Results

Testing samples are obtained using an outdoor overhead camera, and each sample video has a resolution of 320x240. All the experiments are conducted on a 1.3GHz dual-core Intel Core i5 CPU with 4GB of 1600MHz LPDDR3 onboard memory. We have implemented our algorithm in C++ using OpenCV library, for which the source code can be obtained from https://github.com/bvsravikiran/gray. The obtained results are presented in Table 1, and the outputs are shown in Fig 7, 8, and 9.

A. Counting Results and Discussion

To ensure that the results are reliable, we tested our method on various video samples which contain different complex situations, such as, movement of multiple and occluded moving objects; i.e., pedestrians moving in a group. In experiments, our method successfully counted 51 out of 57, obtaining an accuracy of 89.47%.

The algorithm presented in Fig. 5. and Fig. 6. are implemented as a bi-directional counter. In the results(shown in Fig. 7 and 8) the green line at the top used as a reference for counting the objects that are moving downwards, and the green line at the bottom is for the objects moving upwards.

```
loc ← Location of the center of the blob
cSize ← Contour size of the blob
cArea ← Contour area of the blob
regOne ← Region one
regTwo ← Region two
currentState ← 0
MINCAREA ← Minimum contour area a blob should possess to be considered
if cArea > MINCAREA then
    if currentState = 0 and loc = regionOne then
        currentState ← 1
        startTime ← Current system time
    end if
    if currentState = 1 and loc = regionTwo then
        stopTime ← Current system time
        if stopTime − startTime < 2 then
            currentState ← 2
        else
            currentState ← 0
        end if
    end if
    if currentState = 2 then
        UpdateCounter(cSize, cArea)
        currentState ← 0
    end if
end if
```

Fig. 5. GetRegion Algorithm

```
procedure UpdateCounter(cSize, cArea)
    ONE ← Average contour area of a single blob
    TWO ← Average contour area of two blobs
    objectCounter ← Total count of the objects
    while cSize ≠ 0 do
        if cArea = ONE then
            objectCounter = objectCounter + 1
        else if cArea = TWO then
            objectCounter = objectCounter + 2
        else
            objectCounter = objectCounter + 3
        end if
        cSize = cSize − 1
    end while
end procedure
```

Fig. 6. UpdateCounter algorithm

The details of the errors occurred during our hypothesis testing are presented in Table II. There are two common statistical errors in any system; False Positive and False Negative errors. The samples outputs of the errors are presented in Fig. 12. The False Negative error occurred because our system failed to detect the blob. We figured it out that the cause of this error lies in the selection of a number of components in the Image pre-processing stage, and also because the blob might be over-reduced during the erosion operation. However, we the cause

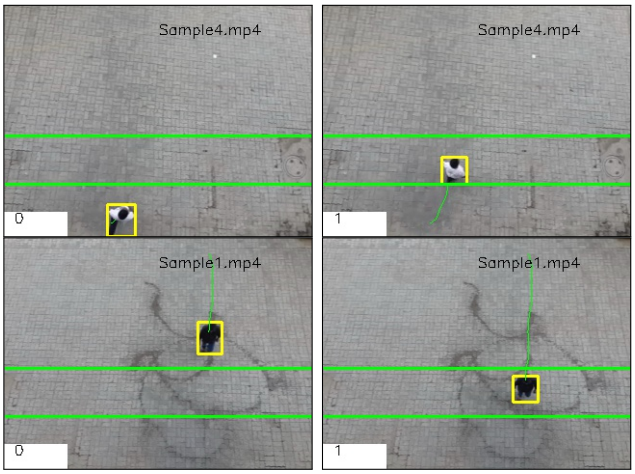

Fig. 7. This figure illustrates that our method achieved object counting. Images at the top indicate a single person moving upwards is being counted, and images at the bottom indicate a person moving downwards is being counted.

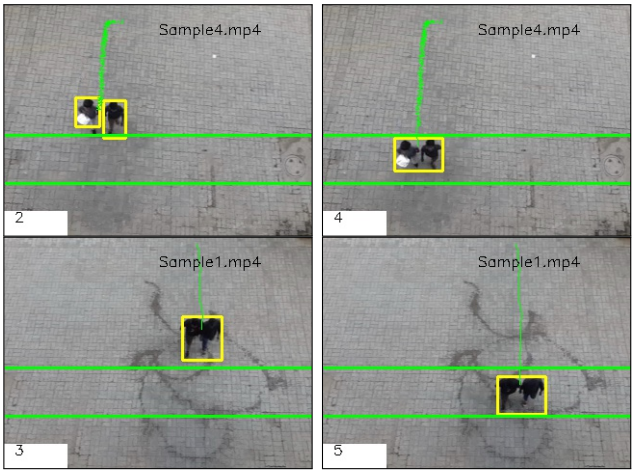

Fig. 8. This figure shows that our method achieved multiple object counting. For example, the images indicate that two people moving downwards are being counted. Note that images at the top and the images at the bottom are taken from different samples.

of the False Positive error is not yet resolved.

B. Tracking Results and Discussion

The tracking results for discussion are shown in Fig. 9. The smoothness of the track largely depends on the process noise or the noise convergence matrix. To achieve a smooth(or an accurate) track with minimal deviation or sudden spikes, we need to set an optimal noise convergence value. Hence, there exists a trade-off between the convergence rate and the track accuracy.

Fig. 9(A) is the outcome of over-hitting, which occurs when the error rate is forced to converge. Therefore, taking a significant value for error rate convergence matrix helps us achieving fast tracking but at a cost of an irregular trail. However, while taking a small value for error rate convergence matrix can obtain us a smooth trail along the path of the object, but may result in slow tracking (as shown in Fig. 9(B)). Therefore, the optimal value for measurement noise convergence is considered as 10^{-3} and the resulted outputs can be seen in Fig. 7 and Fig. 8.

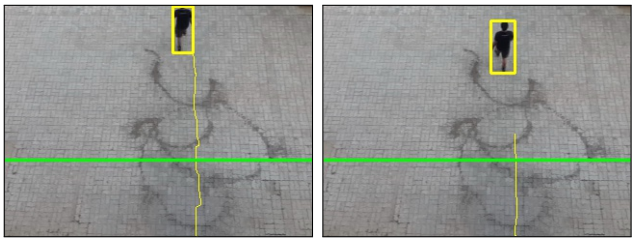

Fig. 9. The figure represents the difference in the smoothness of the trail varied by change in process noise. The image at the top(A) is the result of over-hitting where the value of measurement noise convergence is 10^{-2} and the bottom image(B) represents slow convergence where the value is taken as 10^{-4}.

TABLE I
RESULTS AND ACCURACY PERCENTAGE OF OUR HYPOTHESIS ON EACH OF THE SAMPLES

Sample Name	Ground Truth	Obtained Count	Accuracy%
Sample1	5	5	100%
Sample2	10	10	100%
Sample3	16	13	81.25%
Sample4	26	23	88.46%
Total	**57**	**51**	**89.47%**

TABLE II
ERRORS IN HYPOTHESIS TESTING OF EACH OF THE SAMPLES

Sample Name	False Positives	False Negatives
Sample1	0	0
Sample2	0	0
Sample3	2	5
Sample4	2	5

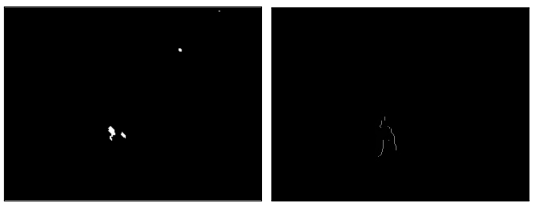

Fig. 10. The figure represents a binary image, which is the output of applying background subtraction algorithm. The image at the left(A) is the resultant blob formed as a result of foreground extraction, and the right image(B) represents detected contours around the blob.

Fig. 11. The figure shows the outputs of morphological operations, applied on Fig. 10(A). The image at the left(A) is the resultant erosion operation, and the right image(B) is the result of Dilation operation.

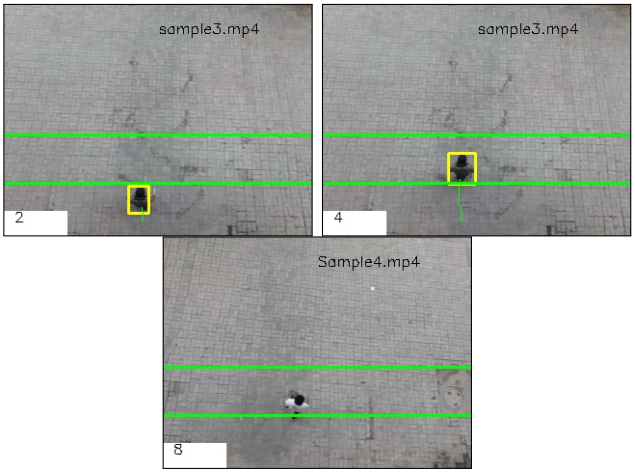

Fig. 12. This figure presents the sample output of the moving object that is not counted by our system. The images at the top show a false positive, where a single object is counted as two. The image at the bottom represents a false negative, where our system was unable to recognize the moving object.

IV. Conclusion

In this paper, we proposed a novel moving-object counting method based on region segmentation and object tracking based on the Kalman Filter. We have used certain heuristics, such as blob dimensions and moving object pace, that makes our method capable of counting multiple objects. Experiments show that our method can obtain a counting accuracy about 89.47%. As a part of future work, the errors can be further reduced by including more heuristics as well as by using a more sophisticated data fusion algorithm.

References

[1] Z. Zivkovic, "Improved adaptive Gaussian mixture model for background subtraction," Pattern Recognition, 2004. ICPR 2004. Proceedings of the 17th International Conference on, 2004, pp. 28-31 Vol.2.

[2] Friedman, Nir, and Stuart Russell. "Image segmentation in video sequences: A probabilistic approach." Proceedings of the Thirteenth conference on Uncertainty in artificial intelligence. Morgan Kaufmann Publishers Inc., 1997.

[3] Rafael C. Gonzalez and Richard E. Woods. 2006. Digital Image Processing (3rd Edition). Prentice-Hall, Inc., Upper Saddle River, NJ, USA.

[4] B. Li, J. Zhang, Z. Zhang and Y. Xu, "A people counting method based on head detection and tracking," Smart Computing (SMARTCOMP), 2014 International Conference on, Hong Kong, 2014, pp. 136-141.

[5] P. Bamrungthai and S. Puengsawad, "Robust people counting using a region-based approach for a monocular vision system," Science and Technology (TICST), 2015 International Conference on, Pathum Thani, 2015, pp. 309-312.

[6] D. Schreiber, A. Kriechbaum and M. Rauter, "A multisensor surveillance system for Automated Border Control (eGate)," Advanced Video and Signal Based Surveillance (AVSS), 2013 10th IEEE International Conference on, Krakow, 2013, pp. 432-437.

[7] Y. B. Brahme and P. S. Kulkarni, "An Implementation of Moving Object Detection, Tracking and Counting Objects for Traffic Surveillance System," Computational Intelligence and Communication Networks (CICN), 2011 International Conference on, Gwalior, 2011, pp. 143-148.

[8] Suzuki, Satoshi. "Topological structural analysis of digitized binary images by border following." Computer Vision, Graphics, and Image Processing 30.1 (1985): 32-46.

[9] Prati, Andrea, et al. "Detecting moving shadows: algorithms and evaluation." IEEE transactions on pattern analysis and machine intelligence 25.7 (2003): 918-923.

[10] Ramer, Urs. "An iterative procedure for the polygonal approximation of plane curves." Computer graphics and image processing 1.3 (1972): 244-256.

[11] Douglas, David H., and Thomas K. Peucker. "Algorithms for the reduction of the number of points required to represent a digitized line or its caricature." Cartographica: The International Journal for Geographic Information and Geovisualization 10.2 (1973): 112-122.

[12] G. R. Bradski, "Real time face and object tracking as a component of a perceptual user interface," Applications of Computer Vision, 1998. WACV '98. Proceedings., Fourth IEEE Workshop on, Princeton, NJ, 1998, pp. 214-219.

[13] Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." Journal of basic Engineering 82.1 (1960): 35-45.

[14] Faragher, Ramsey. "Understanding the basis of the Kalman filter via a simple and intuitive derivation." IEEE Signal processing magazine 29.5 (2012): 128-132.

[15] Greg Welch and Gary Bishop. 1995. An Introduction to the Kalman Filter. Technical Report. University of North Carolina at Chapel Hill, Chapel Hill, NC, USA.

[16] J. H. Lee et al., "Security Door System Using Human Tracking Method with Laser Range Finders," 2007 International Conference on Mechatronics and Automation, Harbin, 2007, pp. 2060-2065.

[17] G. Englebienne, T. van Oosterhout and B. Krose, "Tracking in sparse multi-camera setups using stereo vision," Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on, Como, 2009, pp. 1-6.

[18] C. Fookes, S. Denman, R. Lakemond, D. Ryan, S. Sridharan and M. Piccardi, "Semi-supervised intelligent surveillance system for secure environments," 2010 IEEE International Symposium on Industrial Electronics, Bari, 2010, pp. 2815-2820.

[19] J. Barandiaran, B. Murguia and F. Boto, "Real-Time People Counting Using Multiple Lines," 2008 Ninth International Workshop on Image Analysis for Multimedia Interactive Services, Klagenfurt, 2008, pp. 159-162.

[20] L. Rizzon, N. Massari, M. Gottardi and L. Gasparini, "A low-power people counting system based on a," 2009 IEEE International Symposium on Circuits and Systems, Taipei, 2009, pp. 786-786.

[21] Huazhong Xu, Pei Lv and Lei Meng, "A people counting system based on head-shoulder detection and tracking in surveillance video," Computer Design and Applications (ICCDA), 2010 International Conference on, Qinhuangdao, 2010, pp. V1-394-V1-398.

[22] J. Garca, A. Gardel, I. Bravo, J. L. Lzaro, M. Martnez and D. Rodrguez, "Directional People Counter Based on Head Tracking," in IEEE Transactions on Industrial Electronics, vol. 60, no. 9, pp. 3991-4000, Sept. 2013.