Counting Pedestrians in Crowds Using Viewpoint Invariant Training

Dan Kong, Doug Gray and Hai Tao

Department of Computer Engineering

University of California, Santa Cruz

Santa Cruz, CA 95064

(kongdan, dgray, tao)@soe.ucsc.edu

Abstract

This paper describes a learning-based method for counting people in crowds from a single camera. Our method takes into account feature normalization to deal with perspective projection and different camera orientation. Thus, our system is trained to be viewpoint invariant and can be deployed with minimal setup for a new site. This is achieved by applying background subtraction and edge detection to each frame and extracting edge orientation and blob size histograms as features. A homography is computed between the ground plane and the image plane coordinates for the region of interest (ROI). A density map that measures the relative size of individuals and a global scale measuring camera orientation are also estimated and used for feature normalization. The relationship between the feature histograms and the number of pedestrians in the crowds is learned from labeled training data. The two training methods used in the current system are linear fitting and neural networks. Experimental results from different sites with different camera orientation demonstrate the performance and the potential of our method.

1 Introduction

Estimating the number of people in an image is a practical machine vision task that is gaining popularity in the security and surveillance community. The falling cost of miniature surveillance equipment has led to an increase in the number of cameras deployed for surveillance and safety in public places. The cost of constantly monitoring even a small number of these cameras is extremely high. In most instances the operator is watching for potentially dangerous conditions such as overcrowding and unplanned gatherings.

Many methods currently exist for estimating the size and density of a crowd using image processing techniques [2]. Texture analysis has proven to be a useful feature for identifying different levels of crowd density [4, 5]. Marana has shown that crowd density estimates can be obtained by analyzing an image's Minkowski fractal dimension [9]. Feed forward neural networks have been trained to accurately estimate crowd size in real time with relatively simple image features [1, 8, 17]. Cho has shown that these networks can be trained quickly using a hybrid global learning algorithm [1, 8]. The main shortfall of these methods is the lack of spatial information in the feature model. Since people who are farther away from the camera appear smaller, they generally contribute less to the aforementioned crowd estimates. Paragios and Ramesh have attempted to account for this factor by weighing each pixel based on a simple density estimate [7]. Another technique to count people is based on change detection and

BMVC 2005 doi:10.5244/C.19.63

blob tracking [12, 13, 14]. These systems are mainly aimed at classifying objects and activities by analyzing their shapes and trajectories. They explicitly rely on their capability to separate an object from the background. In scenarios where people are crowded, the system performance is poor since precise object extraction is nearly impossible. Attempts have been made to use existing face and pedestrian detectors for estimating crowd size and density. In [16], a system is developed to detect crowd in real-time by looking at the motion patterns of crowds in the spatial-temporal domain. Lin has proposed using a perspective transform to estimate crowd size using the output of a face detection algorithm [3]. Local features including appearance and motion are also used to detect pedestrians [19]. However detection based methods all rely on the output of the detector, which can be of poor quality under viewpoint or illumination changes and are computationally expensive. There are also systems using multiple cameras to count people [10, 18]. This can alleviate the difficulties that plague single camera systems, such as occlusion. However, deploying such a system is expensive and all the cameras need to be calibrated in advance.

In this paper, we developed a system to count pedestrians in crowds from a single camera and our contributions have two aspects. First, instead of using simple features such as a summation of edge or blob pixels, feature histograms are adopted. This representation is more powerful than simple features in terms of handling noise and can more accurately count pedestrians moving in groups. Second, unlike previous learning-based approaches, our method takes into account the feature normalization to deal with perspective projection and camera orientation. Thus, our system is trained to be viewpoint invariant and can be deployed with minimal setup.

The rest of this paper is organized as follows. Section 2 discusses feature extraction and normalization. Section 3 describes training methods used in our system. In section 4, we give the implementation details and show some experimental results. A conclusion and discussion can be found in Section 5.

2 Feature Extraction and Normalization

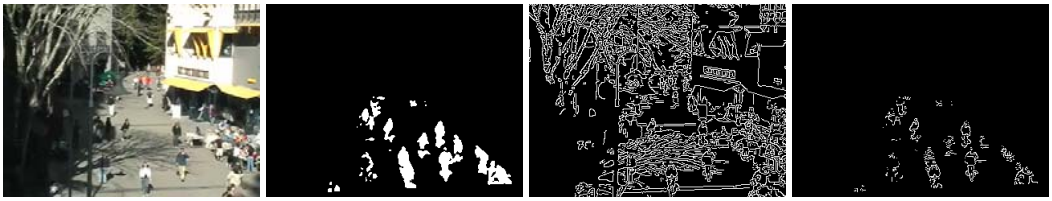
2.1 Feature Extraction

The features include foreground regions given by a background subtraction algorithm and the edge orientation map generated by an edge detector, as shown in Figure 2. The mixture-of-Gaussian based adaptive background modeling method [11] is used to generate foreground mask for each frame as shown in Figure 2(b). Unlike the multiple pedestrian tracking system proposed in [15], we do not explicitly segment the foreground blobs. The reasons are simple. We are only interested in estimating the number of pedestrians, which is easier than tracking. For highly crowded sites the chances of successfully segmenting the foreground blobs to individual people are remote. Instead, we compute a blob size histogram using the foreground mask. Let $H_b(i)$ and $B(i)$ denote the count and size for bin i. The size of a foreground blob k is denoted as $S(k)$. The blob size histogram is formed as follows:

$$H_b(i) = \left\{ \sum_k S(k) \mid B(i) \le S(k) < B(i+1) \right\} \tag{1}$$

This histogram serves two purposes. Firstly, it can model noise resulted from background subtraction. Secondly, the difference between individual pedestrian and group of pedestrians can be captured. For example, a single pedestrian may occupy 100 pixels. When two pedestrians are together, due to occlusion, there will be less than 200 pixels. In other words, each pixel contributes less to the final counts when the blob size is large. The edge detection map is shown in Figure 2(c). We apply canny edge detector to each frame. Figure 2 (d) shows the binary edge map after the AND operation between (c) and the foreground mask (b). In [1], the total edges length is one of the features used to represent

crowd density. In our system, we use an edge orientation histogram instead of a summation of edges. Such a representation can distinguish edges caused by pedestrians, which are usually vertical, with other scene structures such as noise, shadows and cars.



(a) (b) (c) (d)

Figure 2 Features: (a) original image, (b) foreground mask image (c) edge detection map, (d) the edge map after the 'AND' operation between (b) and (c).

As an illustration, the histograms corresponding to Figure 2(b)(d) are shown in Figure 3. For the edge orientation histogram (a), the horizontal axis corresponds to the eight orientations quantized from 0~180 degrees. For blob histograms (b), the horizontal axis represents different blob size. Analyzing the two histograms gives us some intuition into why a histogram based representation is more expressive than raw features, when used in pedestrian counting. For the edge orientation histogram in figure 3(a), we observe that the first and last bins dominate. They correspond to the number of vertical edges in the scene, which is strongly correlated with the number of pedestrians present. For the blob size histograms, we can tell from its 'central peak' peak shape that most of the pedestrians are distributed sparsely in this scene.

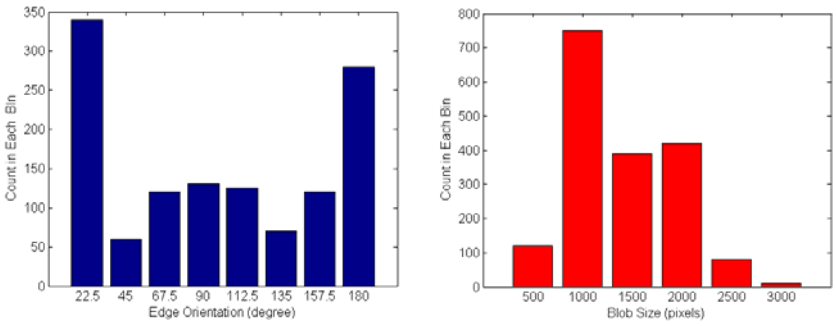


Figure 3. Left: edge orientation histogram. Right: blob size histogram.

2.2 Density Estimation

The output of the density estimation module is a density map associated with the ROI and a scale factor. They are used to normalize the features with respect to object translation and different viewpoints as described in 2.3. Two assumptions are made in our density estimation model. It is assumed that all the pedestrians in the scene have similar size, and that they all lie on a horizontal ground plane.

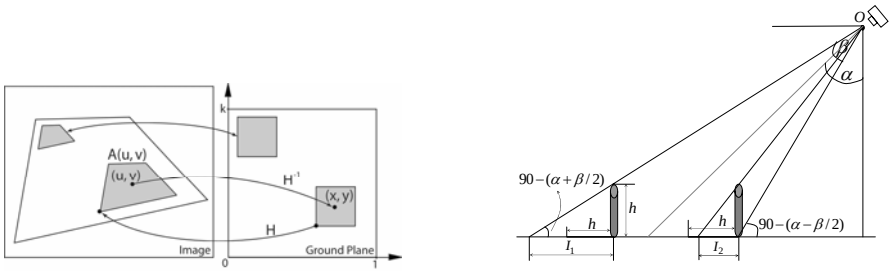
In this paper, a simple cylinder model is adopted for the pedestrians. Due to perspectivity, the projected 2D height varied when the people move on the ground plane. Assume $\mathbf{u_0}$ is the image projection of a point $\mathbf{x_0}$ on the ground plane. Point $\mathbf{x} = [x, y, h]^T$ is vertically above $\mathbf{x_0}$. Then the measured 2D height

is $l = \| \mathbf{u_0} - \mathbf{u}(h) \|$, where $\mathbf{u}(h)$ is the projection of point x. When people translates on the ground, we can measure the ratio of the projected height l_1 / l_2. The ratio reflects the relative crowd density on the ground plane and accurate estimation of it requires camera calibration. In our system, it is approximated by estimating the *homography* between the ground plane and image plane, and we show that this approximation is valid when the camera orientation is not close to an overhead scenario.

The points in the image and the point on a 3D plane are related by a plane perspective transformation called *homography*, represented by a 3 by 3 matrix H. Suppose the approximate ratio of ground plane width and height is k. Given the ROI in the image and the corresponding ground plane as shown in Figure 4(a), we define the origin of the world coordinate at the bottom left corner of the ground plane. Since we only need to estimate the relative density, we do not need the absolute size of ground plane. Let the coordinates for the four corner points be $(0,0),(1,0),(1,k),(0,k)$ respectively. The point $(u,v,1)$ in the image plane is related to point $(x,y,1)$ in the ground plane by H. For each point correspondence, we can generate the following two equations for the elements of H:

$$\begin{aligned} u(h_{31}x + h_{32}y + h_{33}) &= h_{11}x + h_{12}y + h_{13} \\ v(h_{31}x + h_{32}y + h_{33}) &= h_{21}x + h_{22}y + h_{23} \end{aligned} \tag{2}$$

Four point correspondences are sufficient to solve for H up to a scale. By simply manipulating the equations, we write it in a homogenous system $Ah = 0$ of eight equations in nine unknowns. Using *Lagrangian* multipliers, the solution is the eigenvector corresponding to the smallest eigenvalue of matrix $A^T A$.



(a) (b)

Figure 4: (a) *Homography* computation and density estimation for image ROI (left) and ground plane (right). (b) An approximation to the true density map.

Once we estimate the *homography* H between the ground plane and the image, the next task is to compute the relative density $d(u,v)$ for each pixel (u,v) in the ROI. To do this, we first pick a reference point $r(u,v)$ in the ROI. Applying the inverse *homography* leads to the coordinate of the corresponding point $r(x,y) = H^{-1} r(u,v)$ in the ground plane. We pick a square centered at $r(x,y)$ and warp the four corner points of the square back into the image plane using the homography as shown in Figure 4(a). The area covered by the trapezoid in the image plane is computed and denoted as A_r. Then we repeat this process and compute the area $A(u,v)$ for all the pixels inside the region of interest. For each pixel, the relative density is defined as $d(u,v) = A(u,v) / A_r$. This leaves the reference point has density equal

to one. Figure 5 shows the region of interest (ROI) and the geometric weight image associated with it. The darker pixel values in the ROI correspond to more weight. It can be seen that the perspectivity is captured using the quasi-calibration information.

Figure 5: Density estimation. Left: region of interest in the image. Right: density map (weights).

Since the density map computed from *homography* is an approximation to the true density, it is appropriate to discuss under what conditions this approximation is valid. A conclusion can be drawn by analyzing figure 4(b). Let h denotes the standard pedestrian height and the unit length on the ground plane. β and α are the field-of-view and orientation of the camera. We consider the density estimate for pedestrian on the boarder of the FOV, and for simplicity, only one dimension is shown here. The density estimation in our method is to project line segments h onto the image plane using the *homography* and compute their ratio. However, from figure 4(b), we can see that the true density d_{true} computed using projection matrix and the density d we estimated is related by

$$\frac{d_{true}}{d} = \frac{I_1}{I_2} \approx \frac{\tan(\alpha + \beta/2)}{\tan(\alpha - \beta/2)} \tag{3}$$

It can be seen from equation (3) that the density map we estimated will be close to the true one as α increases. However, as the camera goes more and more overhead, our density estimation model will be far away from true density. For the extreme case, when the camera is straight overhead, our estimation method will give a constant estimation.

Another geometric distortion we need to account for is different camera setup. For example, the size of pedestrian given by an overhead camera scenario will be smaller than an oblique camera. To make the training invariant to different viewpoint, we adopt the following strategy. First, the standard height to width ratio of our cylinder model is defined as $k = h/w$. Then, for each training site, when pick the reference point for density estimation, we also measure the height to width ratio of pedestrian in the image at that point. Let it be k', then the scale for that training site will then be approximated as $S = k/k'$. To see why this scale factor captures the different camera orientation, we use a pin hole camera model for illustration. Let h' and w' be the projected height and width in the image. Under the cylinder model, the projected width will not be affected by the camera orientation. Using basic projective equation, the ratio k' is computed as:

$$k' = \frac{h'}{w'} \approx \frac{h \times \cos\theta \times f/z}{w \times f/z} = k\cos\theta \tag{4}$$

Where f is the focal length, θ is the camera orientation and z is the distance from projection center to the people. It can be seen that the scale factor $S = k/k' = 1/\cos\theta$ gives us an indirect measure of the camera orientation.

2.3 Feature Normalization

Given $d(u,v)$ and S, features are normalized with respect to relative density and orientation scale. The purpose of this feature normalization is to give us a measure of the features that is approximately invariant to the translations of pedestrians on the ground plane and under different camera viewpoint. Suppose binary edge orientation and blob map have been extracted from each frame. The edge and blob pixels are normalized with respective to the relative density and scale before constructing the histograms. For the edge orientation map, the normalization is: $or'(u,v) = or(u,v) * S / \sqrt{d(u,v)}$, where $or(u,v)$ is the equal to one for edge pixels and zero for non-edge pixels. $or'(u,v)$ is the value after normalization. By doing this, we assume that the size of edge changes linearly with respect to the density and scale. For the change detection map, the normalization is: $blob'(u,v) = blob(u,v) * S^2 / d(u,v)$. Here we assume that the size of blob changes quadratically. Then the histograms are constructed from the normalized edge orientation and blob map. To show that the feature normalization does give a measure that is approximately invariant to positions of pedestrians on the ground plane and to different camera setup, a simple example is provided, as shown in Figure 6. We consider measuring the features of a single person translating on the ground plane with two different viewing angles.

(a) (b)

Figure 6: A single person example: (a) original video sequence with 30 degree camera orientation. (b) original video sequence with 60 degree camera orientation.

The edge orientation histograms are computed for each frame of the two camera setup. For the ease of display, discrete histograms are interpolated to be smooth. As Figure 7 shows, the feature normalization gives us an invariant measure for the same person translating on the ground plane under two different viewing angles.

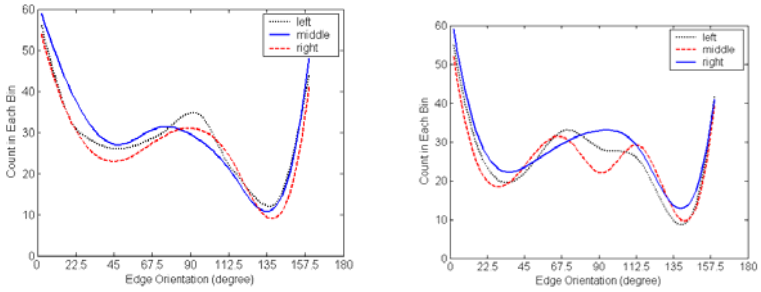


Figure 7: The edge orientation histograms corresponding to the three frames of 30 degree orientation (left) and 60 degree orientation (right) in figure 8.

3 Training

The goal of training is to find the relationship between the features and the number of pedestrian in the image. In our current system, the training is carried out in a supervised way. The pedestrian count is

determined manually for each training frame through a user interface. Two training methods are described in this section: linear regression and neural network.

3.1 Linear Fitting

Let $H_b(i)$ and $H_o(i)$ denote the blob and edge orientation histograms with m and n bins respectively. The pedestrian count estimate is modeled as the linear combination of the two histograms using $m+n$ weights W_b and W_o as follows:

$$C = \sum_{i=1}^{m} H_b(i) * W_b(i) + \sum_{j=1}^{n} H_o(j) * W_o(j) \tag{5}$$

If we stack the two histograms and the corresponding weights into two vectors H and W and let N denote the total number of training frames, we obtain a linear system:

$$\begin{bmatrix} C_1 \\ C_2 \\ . \\ . \\ . \\ C_N \end{bmatrix} = \begin{bmatrix} H_1(1) & H_1(2) & . & . & . & H_1(m+n) \\ H_2(1) & H_2(2) & . & . & . & H_2(m+n) \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ . & . & . & . & . & . \\ H_N(1) & H_N(2) & . & . & . & H_N(m+n) \end{bmatrix} \begin{bmatrix} W_1 \\ W_2 \\ . \\ . \\ . \\ W_{m+n} \end{bmatrix}$$

Since $N \gg m+n$, the linear system is over-determined. We solve it using least square method and the output weights are used online in the system to count pedestrians. One of the main limitations of linear model is that it can not handle occlusions due to its simple assumption of the linear relationship between features and the count of pedestrian. To address this problem, we turn to neural network.

3.2 Neural Network

For single camera pedestrian counting, accurate estimation becomes challenging with significant occlusion. Under such conditions, the pedestrian count and the features size does not have a simple linear relationship. For example, the blob size for a group of pedestrians walking together (highly occluded) would be different from those walking individually. To capture the nonlinear mapping between the input features and the crowd count from the training data, we use a single hidden layer neural network. There is nothing particularly magical about neural networks: they implement linear discriminants in a space where the input have been mapped nonlinearly, which is what we exactly want here. In our system, the neural network model has a single hidden layer. The input layer has $m+n$ input units, which corresponds to $m+n$ bins of blob size and edge orientation histograms. There is only one output unit in our neural network, representing estimation of the crowd count. Bias units are also used in each layer. Sigmoid activation function is used for all the nodes in the network. We train this network using standard back propagation (BP) algorithm with batch presentation. After the training has converged, we read out the weight matrix and bias vectors from the network and use them for online counting.

4 Implementation and experimental results

The crowd estimation system was implemented using OpenCV and DirectShow. During training, we choose camera orientations ranging from horizontal to almost overhead. This covers most of the scenarios in visual surveillance. Currently, the system is running on a PC with 1.7Hz Pentium IV processor, processing a 320x240 video sequence at the rate of 5 frames per second. The system can be applied in real-time with an optimization. Once the training is done, we can deploy the system for online

crowd counting. To train the system and to verify the performance of the proposed method, the true number of pedestrians in crowds should be provided in order to be compared with the result estimated by the system. In our current implementation, the ground truth is obtained manually.

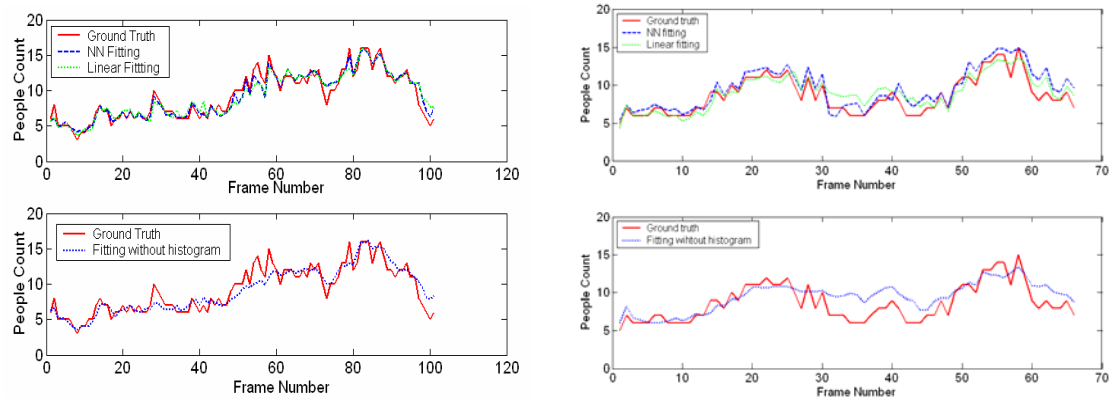
For each training sequence, we first pick the region of interest and specify the reference point in the first frame via the user interface. Then features are extracted and normalized every 20 frames. Histograms are computed based on the normalized features. In our implementation, edge orientation histogram has eight bins quantized from 0 to 180 degrees. For blob size histogram, since picking the optimal bin number and bin size is nontrivial, we choose them empirically. Currently, 6 bins with size 500 and uniform spacing are used.

The trained system is first applied to count crowd from the same site with two different camera orientations, 30 and 70 degrees approximately. The counting is done offline so that we can compare the result with ground truth. 100 and 65 frames are extracted from the video sequences for the two viewing angles. The estimation results using linear fitting and neural network are shown in figure 8(a)(b). For comparison, we also show the counting results by simply using the edge length and total blob pixels. It can be concluded from figure 14 that for this two sequences, the linear fitting has comparable results to neural network. This may due to a lack of overcrowding in the training data. However, both methods have much better results than the one without using histograms. Comparing figure 8(a) and (b), we can see that the estimation performance degrades as the camera orientation increases. As discussed in section 2, this will cause the density estimation to lose accuracy. Figure 8(c) shows the estimation results for another site. The site is more crowded during the time we collected the data. We can see that neural networks have much better estimation results than a linear model for this example.

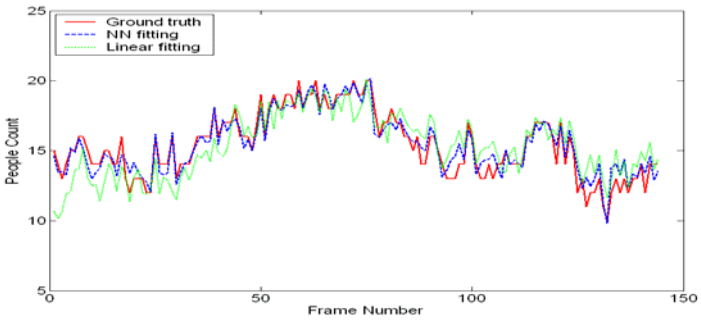
5 Discussions and conclusions

In this paper, a method for counting pedestrians in crowds based on feature extraction and supervised learning is proposed. The main contributions are a novel feature representation and viewpoint invariant feature normalization. Instead of using raw edge and blob features, we use edge orientation and blob size histograms. As a result, noise from background subtraction is suppressed and a nonlinear mapping between the pedestrian count and the feature size due to occlusion is exploited. The viewpoint invariant feature normalization method is proposed to account for distortion introduced by perspectivity and different camera orientation. Our crowd counting system is trained offline in a supervised manner. The training methods adopted in current system include linear fitting and neural network. A patch-based training method is currently being investigated as well. By decomposing the images into patches and train the system based on local features, the performance of the system is expected to be improved. The experimental results from different sites with different camera setup demonstrate the accuracy and reliability of our system.

Even though the system works well for most of our testing data, it is still untested on some complex situations such as when the overlap among pedestrians is large and some persons are completely occluded by others. For this reason, we are currently investigating how to use prior and temporal information to deal with occlusion. Another solution is to use multiple cameras instead of one. Besides highly occluded scenes, the current system is also limited to estimating the size of moving crowds, since our features are based on change detection. In the future, the performance of the system may be improved by incorporating pedestrian or head detector as features. As a result, static pedestrians may be included in the estimate as well.



(a) (b)



(c)

Figure 8: Crowd counting results (a) 30 degree sequences from site A. (b) 70 degree sequence from site A (c) Sequence from site B.

References

[1] S.-Y. Cho, T. W. S. Chow, and C.-T. Leung, "A neural-based crowd estimation by hybrid global learning algorithm," *IEEE Trans. Syst, Man, Cybern. B*, vol. 29, pp. 535-541, 1999.

[2] A. C. Davies, J. H. Yin, and S. A. Velastin, "Crowd monitoring using image processing," *Electron. Commun.* Eng. J., vol. 7 pp. 37 – 47, 1995.

[3] S. -F. Lin, J. -Y. Chen, H. -X. Chao, "Estimation of Number of People in Crowded Scenes Using Perspective Transformation," in *IEEE Trans. System, man, and cybernetics*, vol. 31, No. 6, 2001

[4] A. N. Marana, L. F. Costa, R. A. Lotufo, and S. A. Velastin, "On the efficacy of texture analysis for crowd monitoring," in *Proc. Computer Graphics, Image Processing, and Vision*, 1998, pp. 354-361.

[5] A. N. Marana, S. A. Velastin, L. F. Costa, and R. A. Lotufo, "Estimation of crowd density using image processing," *in Proc. IEE Colloquium Image Processing for Security Applications*, 1997.

[6] D. Biliotti, G. Antonini, J. P. Thiran, "Multi-layer hierachical clustering of pedestrian trajectories for automatic counting of people in video sequences", *IEEE Workshop on Motion and Video Computing*, 2005.

[7] N. Paragios, V. Ramesh, "A MRF-based approach for real-time subway monitoring," In *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2003.

[8] T. W. S. Chow, J. Y. –F. Yam, S. –Y. Cho, “Fast training algorithm for feedfoward neural networks: application to crowd estimation at underground stations,” *Artificital Intelligence in Engineering*, vol. 13, pp. 301-307, 1999.

[9] A. N. Marana, L. da F. Costa, R. A. Lotufo, S. A. Velastin, “Estimating crowd density with Minkoski fractal dimension,” in *Proc. Int. Conf. Acoust., Speech, Signal Processing*, vol. 6, Phonex, AZ, 1999, pp.3521-3524.

[10] V. Kettnaker, R. Zabih, “Counting people from multiple cameras,” *in IEEE International Conference on Multimedia Computing and Systems*, vol. 2, pp. 267-271, 1999.

[11] C. Stauffer, W. Grimson, “Adaptive Background Mixture Models for Real-time Tracking,” In *IEEE Conf. on Computer Vision and Pattern Recognition*, Colorada, USA, 1999.

[12] M. Isard, J. MacCormick, “BraMBLe: A bayesian multiple-blob tracker,” in *ICCV*, v. 2, pp. 34-41, 1999

[13] I. Haritaoglu, D. Harwood, L. S. Davis, “W4: Real-time surveillance of people and their activities,” In *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, No. 8, August 2000.

[14] R. Collins, A. Lipton, T. Kanade, “A system for video surveillance and monitoring,” *American Nuclear Soc. 8th Int. Topical Meeting on Robotics and Remote Systems*, 1999.

[15] T. Zhao, R. Nevatia, and F. Lv, “Segmentation and tracking of multiple humans in complex situations”, *In CVPR01*, 2001.

[16] Reisman, P. Mano, O. Avidan, S. Shashua, A. “Crowd detection in video sequences,” in IEEE *Intelligent Vehicles Symposium*, 2004.

[17] C. S. Regazzoni and A. Tesei., “Distributed data fusion for real-time crowding estimation,” *Signal Process.*, vol. 53, pp. 47-63, 1996.

[18] Danny B. Yang, Hector H. Gonzalez-Banos and Lenidas J. Guibas, “Counting people in crowds with a real-time network of simple image sensors,” In *IEEE International Conference on Computer Vision (ICCV)*, France, 2003.

[19] P. Viola, M. Jones, and D. Snow. “Detecting pedestrians using patterns of motion and appearance”, In *IEEE International Conference on Computer Vsion (ICCV’03)*, France, 2003.