Real-time people counting system using a single video camera

Damien Lefloch[a,b], Faouzi Alaya Cheikh[b], Jon Yngve Hardeberg[b], Pierre Gouton[a] and Romain Picot-Clemente[a]

[a] University of Burgundy, BP 47870, 21078 Dijon Cedex, France;
[b] Gjøvik University College, P.O. Box 191, N-2802 Gjøvik, Norway

ABSTRACT

There is growing interest in video-based solutions for people monitoring and counting in business and security applications. Compared to classic sensor-based solutions the video-based ones allow for more versatile functionalities, improved performance with lower costs. In this paper, we propose a real-time system for people counting based on single low-end non-calibrated video camera.

The two main challenges addressed in this paper are: robust estimation of the scene background and the number of real persons in merge-split scenarios. The latter is likely to occur whenever multiple persons move closely, e.g. in shopping centers. Several persons may be considered to be a single person by automatic segmentation algorithms, due to occlusions or shadows, leading to under-counting. Therefore, to account for noises, illumination and static objects changes, a background substraction is performed using an adaptive background model (updated over time based on motion information) and automatic thresholding. Furthermore, post-processing of the segmentation results is performed, in the HSV color space, to remove shadows. Moving objects are tracked using an adaptive Kalman filter, allowing a robust estimation of the objects future positions even under heavy occlusion. The system is implemented in Matlab, and gives encouraging results even at high frame rates. Experimental results obtained based on the PETS2006 datasets are presented at the end of the paper.

Keywords: Video analysis, video surveillance, background estimation, segmentation, object tracking

1. INTRODUCTION

Knowing the exact number of persons in a building, building-floor, or a single room can be critical for the success of business or rescue operations. Therefore, shopping centers, are required to know the exact number of persons present in their premises, at any point in time. Thus, they often purchase and implemente both people-counting and video surveillance systems. Each of these systems is typically dedicated to a single task; either counting or monitoring the people within a certain area. Even though they are operating within the same area and performing related tasks, they usually do not interact in anyway and thus do not benefit from the information collected by the other system. Additionally, people-counting systems typically estimate the number of people passing through a gate by counting the number of times a beam of light, e.g. infra-red light, is interrupted. Even though it is very simple, such a system can be very efficient in scenarios where no two persons, or objects in industrial production lines, pass through the monitored gate at the same time. They fail, however, to accurately count the number of people passing through a gate of a shopping mall which is typically wide enough to allow several persons to enter at once. Moreover, these systems do not distinguish between passing persons and objects such as carts or baby strollers. For all the above mentioned reasons we have been asked by a Norwegian company to build a scalable system that can accurately count people in a single room or a group of shopping malls, combining the existing beam-based counting systems and video surveillance systems. Therefore, in the first phase of the project we have identified and worked on four complementary parts:

Further author information:
D.L.: E-mail: prince.vladtepes@gmail.com; F.A.C.: E-mail: faouzi@hig.no; J.Y.H.: E-mail: jon.hardeberg@hig.no
P.G.: E-mail: pgouton@u-bourgogne.fr; R.P.C.: E-mail: kithrsb@gmail.com

Real-Time Image Processing 2008, edited by Nasser Kehtarnavaz, Matthias F. Carlsohn,
Proc. of SPIE-IS&T Electronic Imaging, SPIE Vol. 6811, 681109, © 2008 SPIE-IS&T · 0277-786X/08/$18

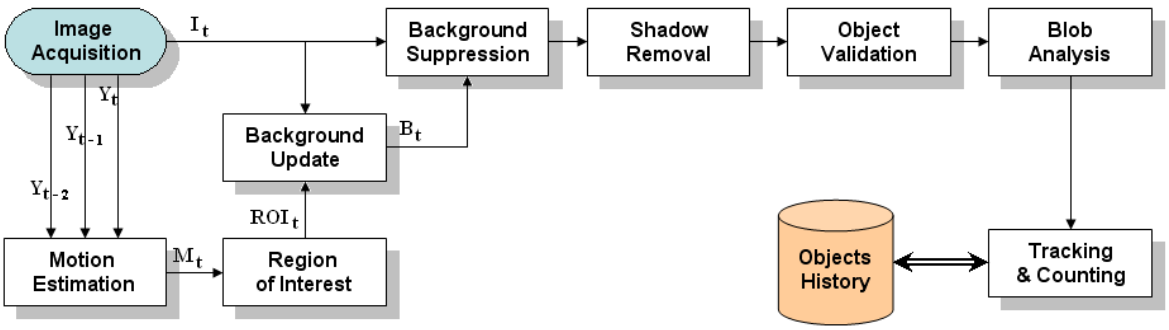

Figure 1. System overview

Several works have been made in this area to obtain more accurate and reliable people-count estimations. An intuitive solution to this problem is to obtain a head count of the persons. While this would be feasible for a human, it is a difficullt task for an automatic system. Lin[1] used wavelets to extract head-shaped features frome the image. Support vector machine processing is then used to classify correctly the feature as `head` or `something else`. A different approach to segment group of people is to use the information of various camera views. The M2-tracker[2] assigns pixels for each camera views using color histograms to a particular person. To avoid the problem of people occlusion, Kim[3]proposed to mount an overhead video camera and obtained an effective count (96%).

In this paper, we present a method for real-time people counting in buildings (indoor scene) using a single static video camera. The paper is organized as follows: Sections 2,3,4 describe the different steps of our proposed system (see Fig. 1). Experimental results are discussed in Section 5. And finally, we conclude this paper with conclusions and future works in Section 6.

2. BACKGROUND ESTIMATION

To detect moving objects in a sequence is to use a simple frame differencing between a background model and current frame can yield satisfactory results in most applications. The main challenge of background substraction is to estimate a robust background of the scene to deal with illumination and static objects changes (or ghosts). The last one occurs, for example, when a static object moves. A standard adaptive background substraction will detect false positive (static object ghost) for a short time which make the tracking process more difficult. Consecutive frames differencing is not subject to this phenomena but cannot detect full moving objects (just highlight the bounds of moving objects). For those reasons, we propose a robust background estimation[4,5] by combining adaptive background generation with three-frame differencing algorithm.

We assume that the video sequence is captured in an RGB color space by a stationary video camera. The problem with the RGB color space is its great sensitivity to sensor noise and changes of lighting conditions. Therefore we compute the luminance component of the color image and use it to estimate the motion for each frame. Let $I_t(x, y)$ be the color value of the pixel (x,y), at time t (the pixels color components values vary in the intensity range of $[0, 255]$). We compute the luminance values Y_t of I_t as a weighted sum of the R_t, G_t, and B_t components; according to the following equation:

$$Y_t(x, y) = 0.2989 * \mathbf{R_t}(x, y) + 0.5870 * \mathbf{G_t}(x, y) + 0.1140 * \mathbf{B_t}(x, y)\,. \quad (1)$$

Only the luminance information will be used in the following to estimate the background model and to update it over time. In this paper, the color information is used for the shadow removal only.

The first frame of the sequence I_0 is used as an initial background estimate B_0. This preliminary background can be totally erroneous if foreground elements are present in the field of view of the video camera but will converge to a robust background in a short period of learning time (when compared to other adaptive algorithms which typically require a long time of initialization[6,7] or need no presence of people during a certain time[8]). A binary motion mask M_t , $t > 1$ is defined by thresholding the two difference frames between each three consecutive frames.

If we assume that, in most of the time, the pixel variations are due to sensor noise from the camera and light fluctuations, we can model the motion estimation as follows:

$$M_t(x,y) = \begin{cases} 1 & \text{if } |\mathbf{Y_t}(x,y) - \mathbf{Y_{t-1}}(x,y)| \geq \mu_{t-1} + \sigma_{t-1} \bigwedge |\mathbf{Y_t}(x,y) - \mathbf{Y_{t-2}}(x,y)| \geq \mu_{t-2} + \sigma_{t-2} \\ O & \text{otherwise.} \end{cases}, \quad (2)$$

where μ_{t-1}, μ_{t-2} and σ_{t-1}, σ_{t-2} represent the means and standard deviations of the pixel-wise absolute differences between the pairs of frames $(\mathbf{Y_t}, \mathbf{Y_{t-1}})$ and $(\mathbf{Y_t}, \mathbf{Y_{t-2}})$.

A pixel (x,y) is considered as moving one if and only if its intensity value has changed between the current image and the two previous ones. Therefore, M_t highlights the regions with changes generated by moving objects or illumination.

There is still a problem with this moving object detection: only the pixels that changed between the three consecutive frames are detected thus the moving objects are not totally detected due to the overlap between the body of the moving object in the three frames. Indeed, the algorithm highlights the different edges of moving objects and is unable to detect the whole object (see Fig. 2(b)). If this binary motion mask is used in order to update the background model, then an over-estimation of the background is created (i.e. foreground pixels will be labeled as background pixels; see Fig. 2(c)). Consequently, the segmentation process will be more difficult and could lead to mistakes due to presence of ghosts in the background model. To avoid the problem of over-estimation of the background (i.e. under-estimation of the binary motion mask), we decided to use a regions of interest mask of the binary motion image. First, an object regions labeling is performed on the motion mask (we developed our own algorithm because we found the functions `bwlabel` and `regionprops` of Matlab too slow for real-time computation). Then, some statistics are calculated for each object region: areas (number of pixels in the region) and bounding boxes (the smallest rectangle which completely contains the region). The area is used to filter-out noise by deleting all the objects which have an area smaller than a given threshold τ_{Area} (for our test, the threshold was fixed to 10). While, the bounding box is used to create the ROI_t mask (any pixel belonging to one of the bounding boxes is considered a pixel with motion). Note that the ROI mask is used only to update the background model and never to segment moving objects.

The new background B_t is then computed as a linear combination of the old background B_{t-1} and the current frame I_t in the following way:

$$B_t(x,y) = \begin{cases} \alpha \cdot \mathbf{B_{t-1}}(x,y) + (1-\alpha) \cdot \mathbf{I_t}(x,y) & \text{if } \mathbf{ROI_t}(x,y) = 0 \\ \mathbf{B_{t-1}}(x,y) & \text{otherwise.} \end{cases}, \quad (3)$$

where $\alpha \in [0,1]$ is the learning rate and controls the background adaptation speed. Note that, each pixel (x,y) is updated only if it was classified as non-moving (i.e. $ROI_t(x,y) = 0$). We can simplify Eq. 3 by using the motion mask $ROI_t(x,y)$ and its complement $\overline{ROI}_t(x,y) = 1 - ROI_t(x,y)$:

$$\begin{aligned} B_t(x,y) &= \mathbf{B_{t-1}}(x,y) \cdot \mathbf{ROI_t}(x,y) + [\alpha \cdot \mathbf{B_{t-1}}(x,y) + (1-\alpha) \cdot \mathbf{I_t}(x,y)] \cdot \overline{\mathbf{ROI}}_\mathbf{t}(x,y) & (4) \\ &= \alpha \cdot \mathbf{B_{t-1}}(x,y) + (1-\alpha) \cdot \mathbf{I_t}(x,y) \cdot \overline{\mathbf{ROI}}_\mathbf{t}(x,y) + (1-\alpha) \cdot \mathbf{B_{t-1}}(x,y) \cdot \mathbf{ROI_t}(x,y)\,. & (5) \end{aligned}$$

The variable α determines the update sensitivity to the variations. Obviously, this learning rate is the key parameter to avoid the problem of illumination changes. But, in many applications which use adaptive background subtraction method, this parameter is viewed as empiric and depends on the situations. However, an automatic but simple way to estimate α[5] is to use the rate of motion in the ROI frame, given by the following equation:

$$\begin{aligned} \alpha &= \frac{\text{Number of all moving pixels}}{\text{Total Frame area in pixels}} & (6) \\ &= \frac{\sum \mathbf{ROI_t}}{Area(\mathbf{I_t})} & (7) \\ &= mean(\mathbf{ROI_t})\,. & (8) \end{aligned}$$

Indeed, the more foreground objects are present between the three frames (high presence of moving pixel in the ROI_t mask) the lower the influence of the current image on the background model will be (i.e. α is close to

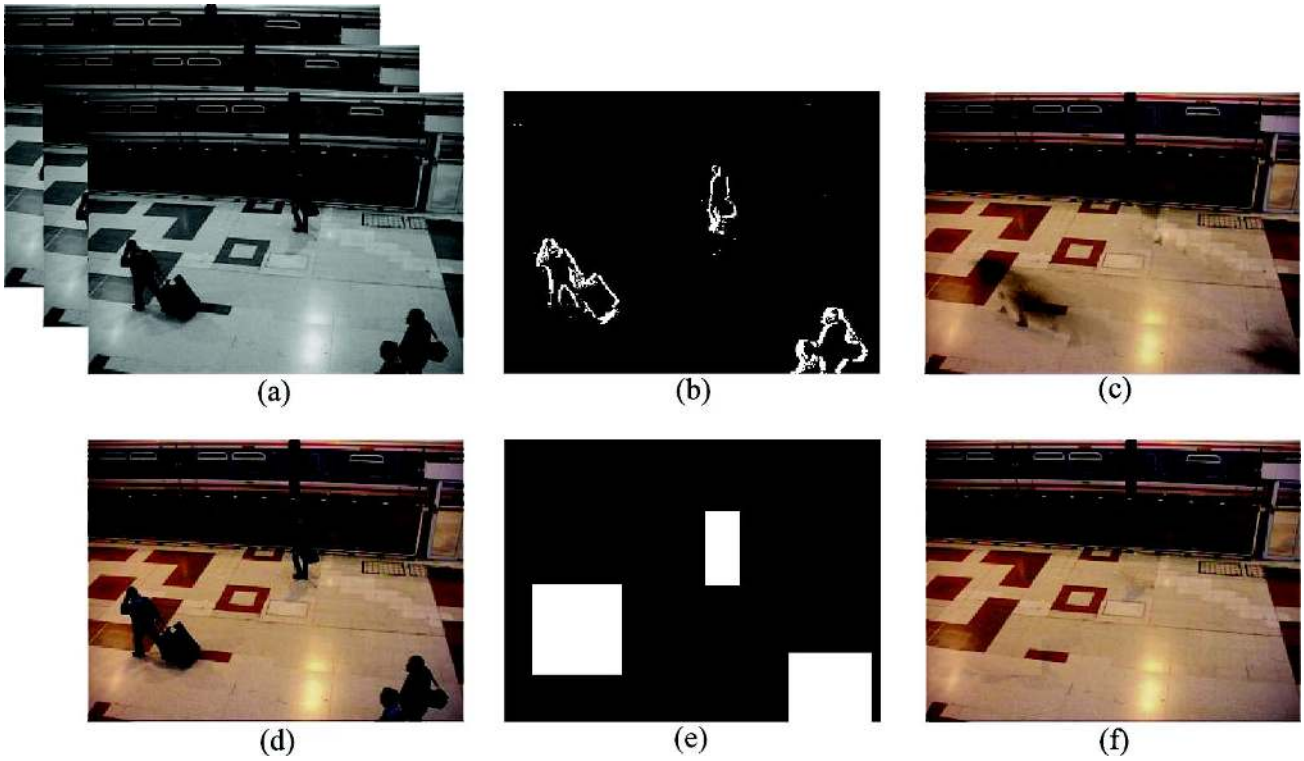

Figure 2. (a) Three consecutive luminance frames; (b) Binary motion mask computed using the frame differencing algorithm; (c) Background model updated with the binary motion mask; (d) Frame #825 (current image); (e) ROI mask of the binary motion mask; (f) Background model updated with the ROI mask.

1). And conversely if the presence of changes is low between the three frames, then the adaptive background is less influenced by the previous background (i.e. α is close to 0). Estimating α automatically gives our algorithm a powerful advantage and enable it to manage with most severe illuminations and objects motions conditions.

Fig. 2 (c) and (f) show the background models estimated using the two updating methods: (c) updated based on the binary motion mask directly and (f) updated based on the ROI mask. In (c) foreground pixels (ghost traces) are present due to the bad estimation of the moving regions. Therefore, the ROI mask method will be used in the rest of our paper in order to update the background model based on real static regions of the frames (as in (f)). The foreground objects detection is then done by a simple subtraction from the adaptive background model.

3. SEGMENTATION

3.1 Background subtraction

Once we have a robust estimate of the background model, we can use it to segment each frame into foreground and background objects. This method is called background subtraction and is a common method to separate background and foreground of a sequence from a stationary camera; the results are relatively good and can be done in real-time.[5,6]

Ideally, a pixel would be part of the foreground, when its value is different enough from its corresponding value in the background model. The main difficulty is to evaluate the distance of each pixel in a color frame (in RGB color space) to the corresponding background pixel. This evaluation allows the classification of all the current image pixels in two categories (foreground and background). In some situations, an oversimplification of the method (for example, an arbitrary definition of a threshold value) may cause erroneous segmentation, and consequently makes the tracking process harder or may even fail. To avoid those drawbacks, an automatic approach is proposed below.

A new RGB color image BS_t, resulting of the background subtraction operation, is created. For each color channel c (R, G or B), an absolute difference is performed between the current frame I_t and the background model B_t.

$$BS_t^c(x,y) = |\mathbf{B_t^c}(x,y) - \mathbf{I_t^c}(x,y)|, \forall c \in \{r,g,b\}\,. \tag{9}$$

Then a new binary mask FG_t is performed to extract the foreground regions and is evaluated as follows:

$$FG_t(x,y) = \begin{cases} 1 & \text{if } \mathbf{BS_t^r}(x,y) > \tau^r \vee \mathbf{BS_t^g}(x,y) > \tau^g \vee \mathbf{BS_t^b}(x,y) > \tau^b \\ O & \text{otherwise.} \end{cases}, \tag{10}$$

where τ^r, τ^g and τ^b are the automatic thresholds for each channel c and are evaluated by analyzing the background-subtracted image BS_t.[4,5,9] We determine the median $MED^c = med(BS_t^c)$ and the median absolute difference $MAD^c = med(BS_t^c - MED^c)$. Supposing that there is motion in less than half of the image, the median parameter MED^c should correspond to a typical noise value of the channel c, thus we define a suitable threshold τ^c (used in Eq. 10):

$$\tau^c = MED^c + 3 \cdot 1.4826 \cdot MAD^c\,, \tag{11}$$

where 1.4826 is the normalization factor for a Gaussian distribution.

In most cases, a simple thresholding is not sufficient to obtain clear foreground regions. Some morphological operations are used to clean up noise (combination of two basic operations `dilatation` and `erosion`). Dilatation has the effect of expanding the foreground and, conversely, erosion expands the background. An erosion followed by an identical dilatation (with the same structuring element) is called an `opening` and is used to eliminate isolated foreground pixels. Thus, to deal with the problem of noise generated by the background-subtracted image (some background pixels can be mis-classified as foreground pixels and vice-versa), a morphological opening followed by a `closing` are performed: the closing fills the missing foreground pixels and the opening removes small isolated foreground ones.

3.2 Shadow removal

Following the segmentation, we could observe that shadows are generally misclassified as moving objects. Indeed, shadows change consequently the color properties in the RGB color space (make darker the color will cause a big variation in the three RGB channels), so the background subtraction method detects shadows as foreground pixels. Even if it is a good thing because the under-segmentation error is reduced, a major difficulty is involved. The problem is that moving shadows are not distinguished from real moving objects, and so could mislead the future tracking module (shadows increase the area of moving object and could even be detected as a new moving object, see Fig. 3.c). Thus, we need to perform a shadow removal operation after the segmentation in order to ensure a reliable tracking process.

Shadows can be interpreted as semi-transparent regions in which the scene reflectance undergoes a local attenuation. So, it is feasible to identify those shadow regions by analysis of their photometric properties. Thus, we have chosen the Hue-Saturation-Value (HSV) color space to explicitly separate chromaticity and luminosity[10,11] which is not possible in the RGB color space and also since it better correlates with the human visual system. The aim is to estimate how the H, S and V values change in the presence of shadows.

In the HSV color space, a shadow and non-shadow points differ principally in the luminance axes V. In order to model the reflectance attenuation of the pixel (x,y), the ratio between its luminance in the current frame I_t^V and its luminance in the background B_t^V must be less than one (i.e. $I_t^V < B_t^V$; shadow has the effect to darken the color). Due to noise and color conversion simplification problems, the chrominance parameters (H and S: hue and saturation channels) will also change, so it is necessary to take into account these variations.

We applied the shadow detection to points belonging to moving objects only, in order to reduce the computational costs and avoid fixed shadow pixels belonging to the background. First, we convert the frames from RGB to HSV color space.

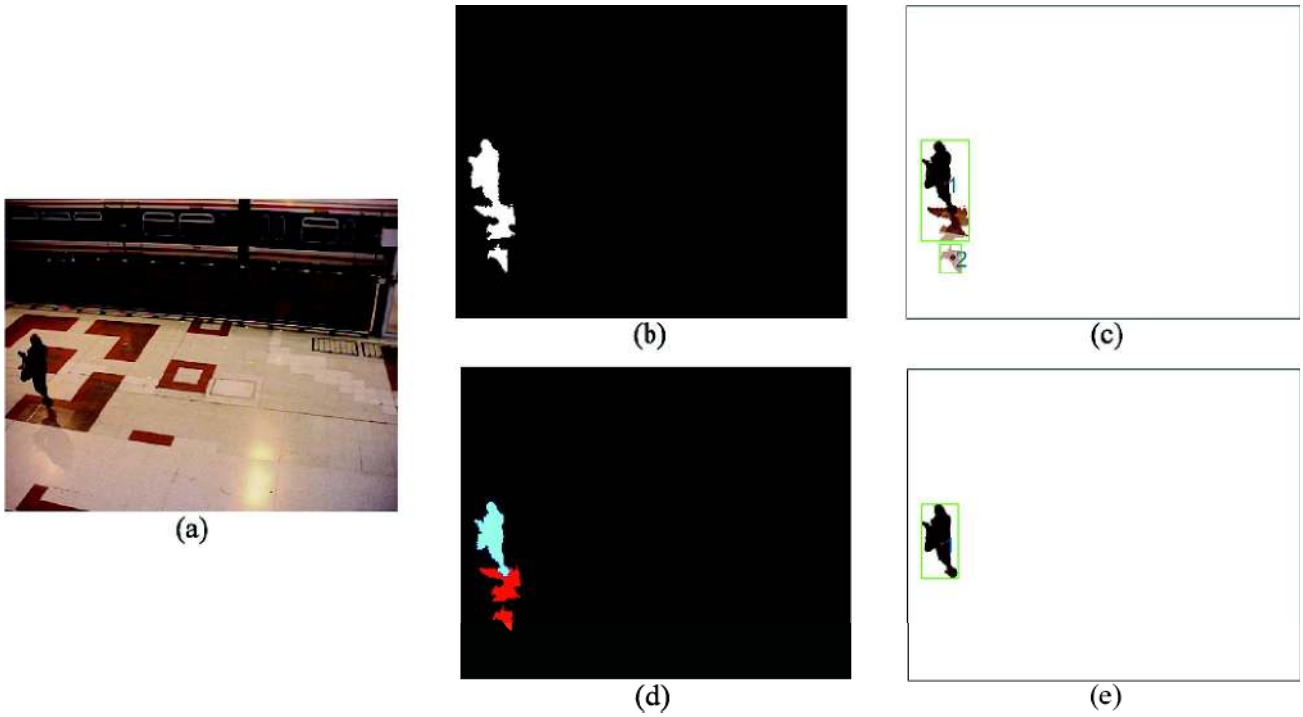

Figure 3. (a) Frame #180 (current image); (b) FG_{180} Binary foreground mask; (c) Foreground image extraction using foreground mask (b); (d) Shadow detection, in light-blue the foreground mask without shadow (dark-red); (e) Foreground image extraction without shadows.

Later, we define a shadow mask SM_t for each point belonging to the foreground based on the following conditions:

$$SM_t(x,y) = \begin{cases} 1 & \text{if } \alpha \leq \frac{\mathbf{I_t^v}(x,y)}{\mathbf{B_t^v}(x,y)} \leq \beta \wedge |\mathbf{I_t^s}(x,y) - \mathbf{B_t^s}(x,y)| \leq \tau_s \wedge \mathbf{D_t^h}(x,y) \leq \tau_h \\ 0 & otherwise. \end{cases} , \tag{12}$$

Where D_t^h represents the `angular` difference between the hue channel of the current image I_t^h and the background B_t^h and is defined as follows:

$$D_t^h(x,y) = min[|\mathbf{I_t^h}(x,y) - \mathbf{B_t^h}(x,y)|, 360 - |\mathbf{I_t^h}(x,y) - \mathbf{B_t^h}(x,y)|] . \tag{13}$$

Two thresholds ($\alpha, \beta \in [0,1]$) are necessary to evaluate the effect of shadow in the luminance channel V. The lower bound α defines a maximum value for the darkening effect of shadow and obviously is proportional to the light source intensity (the higher the light source intensity is, and the lower α has to be chosen). Typically, in normal lighting condition, α ranges from 0.7 to 0.8. And the upper bound β is used to prevent those pixels classified as shadows where the background was darkened too little compared to the shadow effect. Its range is typically from 0.9 to 0.98. The two others conditions (τ_s and τ_h) correspond to the chrominance and saturation channels and are not crucial for the detection of shadows (their value is normally relatively small because of the low effect of shadows on these channels). In the literature, it was found that all these thresholds have an empirical dependence on scene luminance parameters such as the average image luminance and gradient.[12]

Fig. 3 shows the importance of eliminating shadows from the binary foreground mask for the blob analysis algorithm. We can see in Fig. 3 (b) that our proposed segmentation method classifies shadows as foreground objects, consequently objects could grow and shadows could even appear as new objects as can be seen in Fig. 3(c). This problem is resolved using our shadow removal method, see Fig. 3 (e).

3.3 Connected component labeling

To finish the foreground extraction, a fast binary connected component labeling is performed to find the different foreground regions. Our algorithm is based on the algorithm proposed by Haralick and Shapiro[13] and is able to calculate in the same time important statistics for the tracking module. The idea is to scan the binary image along columns and make a primary label image by looking at the pixel's neighbors previously visited and make also an equivalence table containing pairs of connected labels (i.e. referring the same object). A set of features is calculated during this process for each detected foreground object:

- **Areas** : number of foreground pixels for each object. Note that only objects having their area above a certain threshold τ_{area} (fixed to 200 for our tests) are kept for the tracking (to eliminate small objects).
- **Centroids** : coordinates of the center of gravity of each object. It is equal to the mean of all the foreground pixel coordinates composing the whole object.
- **Bounding Boxes** : smallest rectangle which completely contain the object. The upper-left corner (respectively bottom right corner) is equal to minimum (respectively maximum) of all the foreground pixel coordinates of the object.

These statistics are later used in the tracking algorithm.

4. TRACKING AND COUNTING

Tracking an object in a video sequence is the process of finding the same object in different frames. It uses the different features previously extracted by the segmentation module. The rest of this section describes the tracking algorithm we used. It is based on the motion model proposed by Wan[14] (Kalman filter), to predict the future state of every objects in the next frame.

4.1 Motion Model

In real life video, it is safe to assume that movement objects change slowly between two consecutive frames. The object parameters are modeled by a discrete-time kinematic model. Kalman filter is used to predict the state of the object, it is based on the estimation theory. Kalman filter provides a recursive solution where each updated estimate of the state is computed from the previous estimate and the new input data, so only the previous estimate requires storage. The Kalman filter can be used as follows :

$$X_t = A \cdot X_{t-1} + W_{t-1}, \tag{14}$$

$$Y_t = C \cdot X_t + V_t, \tag{15}$$

Where X_t is the state vector, it is defined as the sufficient minimal set of data to describe the unforced dynamical behavior of the object at the time t. W and V represent respectively the state and the measurement noises. They are assumed to be independent, white, and with normal probability distributions. In practice, the state noise covariance matrix Q and measurement noise covariance matrix R might change with each time step or measurement, however here we assume they are constant. W is assumed to have a Gaussian distribution and Q is set as $Q = 0.01 \cdot I$. V can be estimated directly from the data. Y_t is the measurement vector at the time t, C is the observation matrix and A the state matrix. X_t and Y_t are set as follows:

$$X_t = \begin{bmatrix} x(t) \\ y(t) \\ a(t) \\ v_x(t) \\ v_y(t) \\ v_a(t) \end{bmatrix}, \qquad Y_t = \begin{bmatrix} x(t) \\ y(t) \\ a(t) \end{bmatrix},$$

where the elements of X_t represent the centroid coordinates of the object, its area and their corresponding change velocities. So, we can deduce the matrix A and C from the two previous definitions:

$$A = \begin{bmatrix} 1 & 0 & 0 & \Delta t & 0 & 0 \\ 0 & 1 & 0 & 0 & \Delta t & 0 \\ 0 & 0 & 1 & 0 & 0 & \Delta t \\ 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}, \qquad C = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \end{bmatrix}.$$

The Kalman filter has two distinct phases: Prediction and Updating. The prediction phase uses the state estimate from the previous time-step to produce an estimate of the state at the current time-step. In the updating phase, measurement information at the current time-step is used to refine this prediction to arrive at a new more accurate state estimate for the next time-step.

The notation $\hat{X}_{n|m}$ represents the estimate of X at time n given the previous estimate at time m.

Prediction

Predicted estimate state:

$$\hat{X}_{t|t-1} = AX_{t-1|t-1}\,. \tag{16}$$

Predicted estimate covariance (to estimate accuracy of the state estimate):

$$P_{t|t-1} = AP_{t-1|t-1}A^T + Q_{t-1}\,. \tag{17}$$

Updating

Innovation covariance:

$$S_t = CP_{t|t-1}C^T + R_t\,. \tag{18}$$

Optimal Kalman gain:

$$K_t = P_{t|t-1}C^T S_t^{-1}\,. \tag{19}$$

Updated estimate state:

$$\hat{X}_{t|t} = \hat{X}_{t|t-1} + K_t(Y_t C\hat{X}_{t|t-1})\,. \tag{20}$$

Updated estimate state:

$$P_{t|t} = P_{t|t-1}K_t CP_{t|t-1}\,. \tag{21}$$

The last two variables represent the state of the Kalman filter: $\hat{X}_{t|t}$ is the estimate of the state at time t and $P_{t|t}$ its error covariance matrix. In our case, we have the prediction of position and size for objects with an estimate of the error. This prediction is used to build the tracking matrix.[14]

4.2 Tracking Matrix

For each frame, a tracking matrix is built. This matrix links the predicted previous objects position and area to the new ones. The rows of the tracking matrix represent all objects in the current frame, and columns all estimated objects from the previous frame. So, the tracking matrix is a $n \times m$ matrix noted M with n and m indicate, respectively, the number of objects in the current frame and in the previous frame.

Every elements y_{ij} of the matrix is the Euclidean distance between the i-th measurement and the estimated position predicted from the j-th previous object.

$$y_{ij} = \sqrt{(Y_t^i - \hat{Y}_{t|t-1}^j)^T \cdot (Y_t^i - \hat{Y}_{t|t-1}^j)}\,, \tag{22}$$

Where Y_t^i is the i-th measurement in frame t and $\hat{Y}_{t|t-1}^j$ is the j-th estimated object in frame t from frame $t-1$.

$$Y_t^i = (x_i(t), y_i(t), A_i(t)) \,. \tag{23}$$

Note that the y_{ij} elements are recorded if and only if they do not exceed an arbitrary threshold (maximum distance). Beyond this threshold, we assume there is no connection between the considered objects. The tracking matrix is then passed to the Matching Merging and Splitting (MMS) module.

4.3 Matching, Merging and Splitting Module

The MMS module is an ambiguous situation resolver system that permits to determine in which situation the objects are. The first step consists in scanning the tracking matrix along rows and to built an another matrix (called flag matrix). If there is just one non-zero element in the i-th row, then a `splitting or matching` flag is stored. Whereas, if there are more than one non-zero elements in the i-th row, a `merging` flag is stored. Note that if there are only zero elements in the i-th row, then the i-th object is considered to be a new one. The second step consists in scanning this new flag matrix along columns. If there is one `splitting or matching` flag in the j-th column (at the i-th row), then we are sure that it is exactly a `matching` flag (between the j-th previous object and the i-th current object). Note that if there are only zero elements in the j-th column, then the j-th previous object has disappeared from the scene. Otherwise, the flag is totally ambiguous (`splitting, merging or matching`) and thus needs more analysis (find the best distance of all possible combinations between objects; minimize the error of decision). The third step consists in resolving those ambiguous cases and is described below.

- All the possible objects combinations of the j-th column are compared with the j-th previous object. These comparisons are done by calculating the distance between the center of mass of all centroids (from the combination) and the centroid of the j-th previous object. The center of mass is calculated by averaging all the centroids weighted by their corresponding area (see Fig. 4). The final distance equals the sum of the center of mass distance and the area distance. The minimum distance of all the distances computed previously is kept in memory.
- If the j-th column contains `merging` flags, then we also compare all the possible objects combinations of every joined rows containing the j-th object. This is exactly the same method as above.
- Afterwards, only the minus value of every distances is stored. For this value, we have the corresponding combination which is the best objects combination linked with the j-th column.
  - If it does not exceed a certain threshold, all the elements of the j-th column which have no relation with the combination are set to zero in the flags matrix. Moreover, if the combination is on a row, all the elements of the row which have no relation with the combination are also set to zero.
  - Otherwise, the combination cannot be linked with the j-th previous object, so every elements of the corresponding column are set to zero.

Finally, we obtain a perfect flags matrix and we are able to resolve the previous ambiguous cases:

- If there are more than one element on a row, it is a merging situation between previous objects.
- If there are more than one element on a column, it is a splitting situation indicating that the previous object became several objects in the current frame.
- If there is only one object on the row and the column joined, it is a matching situation.

As a result of the MMS module, labels are attributed to every current objects considering their connection with the previous ones.

Fig. 5 shows our proposed tracking method in particular situations. In this video sequence, two persons occlude each other and then go away from each other. In (a), the two persons have exactly two distinct labels (1 and 3). During occlusion ((b): merge situation), their labels are attributed to the resulting blob. Finally, we can see in (c) that the label redistribution is correct after the previous blob splitting.

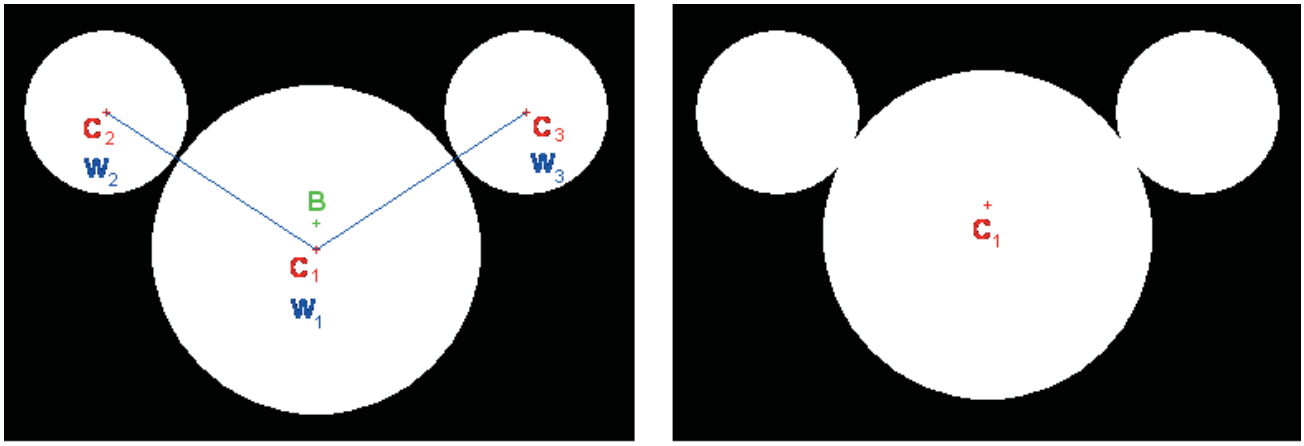

Figure 4. The centroid of the merge resulting object at time t is roughly equal to the center of mass of each object (weighted by their area) at time $(t-1)$ which would compose it. For a split situation, the reasoning is logically reversed.

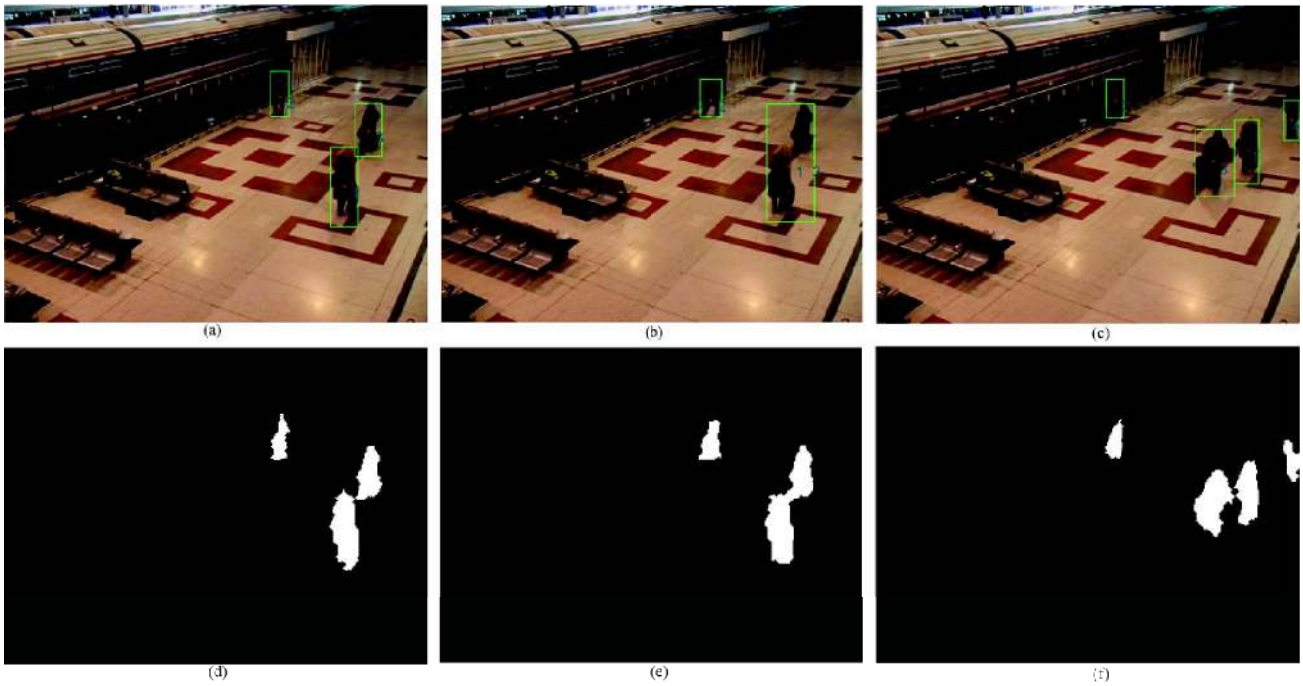

Figure 5. (a),(b),(c): Three frames at different times (#839, #840, #905); (d),(e),(f): Corresponding binary masks.

4.4 Counting

Once we are able to track people, the counting process is relatively easy. Two areas (IN and OUT) are delineated by a virtual line (arbitrary defined). Each time the centroid of an object crosses the line (state modification: transit from an area to the other), the counter linked to the crossing direction is incremented. Note that if an object has multiple labels (due to a merging), the counter is incremented bye the same number of labels.

5. EXPERIMENTAL RESULTS

The current prototype has been developed in Matlab using image processing and image acquisition toolboxes. But, for real-time issue, we decided to implement algorithms in C (Mex files) and compiled them into C libraries as an executable for standard PCs with Microsoft Windows. All our tests were done with the PETS 2006

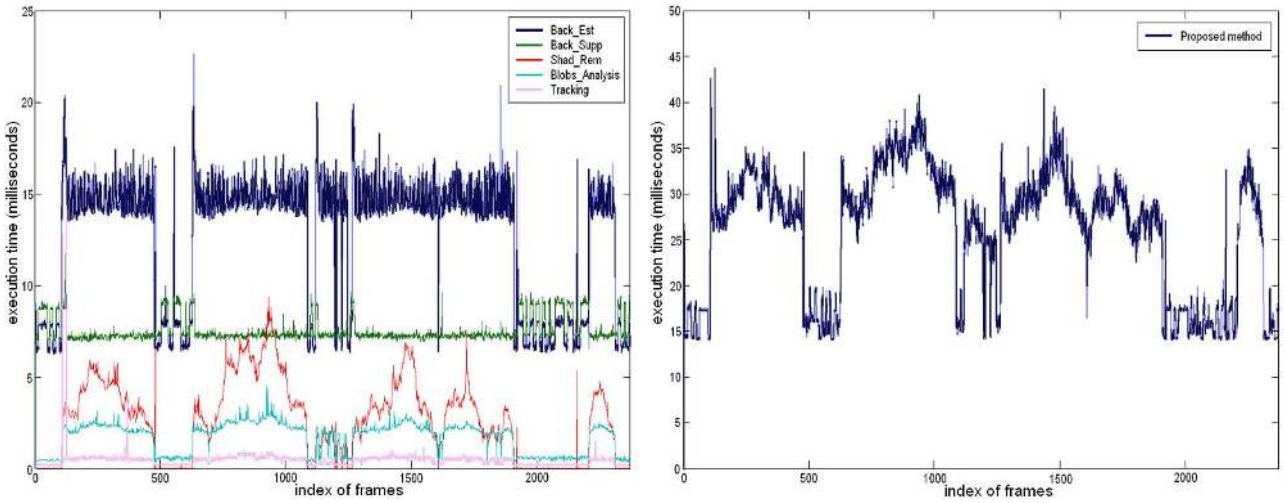

Figure 6. Processing time required for each main algorithms during an entire video stream. The second graph shows the results of our proposed method.

Dataset video streams* at a reduced frame size `320x240` in RGB colorspace. Our current prototype is also effective with intensity video streams but couldn't be as reliable as RGB ones due to the absence of Shadow detection and Removal (see Sec. 3.2 for the reasons). To make our tests, we use a DualCore Intel Centrino 1.66 GigaHertz laptop with 1GBytes RAM. Fig. 6 shows the performance of our proposed system using one video stream (`PETS2006 S3-T7-A-3`). Some statistics are deduced from this figure (See Tab. 1). We can see that the most costly algorithm in computing is the estimation of the background. Indeed we grant to this algorithm much importance because it is determinant for the next evaluations. Thanks to this table, we can also see that our prototype is able to count people at a very high framerate (so some modules can be added to improve our system and make better counts).

Table 1. Processing speed analysis (in millisecond per frame).

	Back. Est.	Segmentation	Shadow Removal	Blob Analysis	Tracking	Prop. meth.
Maximum	21.7	12.3	10.4	9.1	14.4	43.8
Mean	13.2	7.5	2.9	1.8	0.4	25.9

6. CONCLUSIONS AND FUTURE WORK

We present a method to track and count people in complex scenarios at high framerate: such as background changes or crowds moving out or in together. Our proposed system resolves relatively well various troublesome situations such as shadows and ghosts. Shadows are detected using color information and automatically removed. The first improvement of our system could be improving the shadow detection with texture informations[5] in order to enhance it with gray-level image sequences. Ghosts are also totally removed to avoid a further bad segmentation. Actually, common background suppression algorithm is effective in scenes with constant motions. Howewer, if the scene is more complex such as objects stopping and starting their motion, standard techniques will fail. Our proposed system will totally remove the ghost created by the starting moving object with short period of time (initially, the moving object will be connected to its ghost). We can say that our proposed system performs relatively well especially in situations where traditional people counting systems fail: such as crowds moving out or in simultaneously. But our current version is not able to recognize humans in the scene so it can sometimes lead to some erroneous count. The second improvement could be implementing a human recognition

*Found on http://www.cvg.rdg.ac.uk/PETS2006/data.html

for each blob using human motion model or head detection[8] in order to improve the counts. Additionally, such systems have the advantage of allowing for more functionalities at low additional costs therefore making them more cost effective.

ACKNOWLEDGMENTS

This work was carried out with the Norwegian Colour Research Laboratory in collaboration with the Norwegian company P.I.D. Solutions. We would like to thank Pr. El Bay Bourennane from the University of Burgundy for his help in Kalman Filter.

REFERENCES

1. S. F. Lin, J. Y. Chen, and H. X. Chao, "Estimation of number of people in crowded scenes using perspective transformation," in *Transactions on Systems, Man and Cybernetics*, *Proc. IEEE*, pp. 645–654, 2001.
2. A. Mittal and L. S. Davis, "M2tracker: A multi-view approach to segmenting and tracking people in a cluttered scene using region-based stereo," in *The Seventh European Conference Computer Vision*, pp. 18–36, 2002.
3. J. W. Kim, K. S. Choi, B. D. Choi, and S. J. Ko, "Real-time vision-based people counting system for the security door," in *International Technical Conference On CircuitsSystems Computers and Communications*, 2002.
4. R. Collins, A. Lipton, T. Kanade, H. Fujiyoshi, D. Duggins, Y. Tsin, D. Tolliver, N. Enomoto, and O. Hasegawa, "A system for video surveillance and monitoring," Tech. Rep. CMU-RI-TR-00-12, Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, May 2000.
5. A. Leone and C. Distante, "Shadow detection for moving objects based on texture analysis," *Pattern Recognition* **40**, pp. 1222–1233, April 2007.
6. J. V. Huis, "Unsupervised motion segmentation in video images," research assignment, ICT-Delft University of Technology, January 2007.
7. T. Kim, S. Lee, and J. Paik, "Evolutionary algorithm-based background generation for robust object detection," in *Lectures Notes in Computer Science, LNCS 2006*, **4113**, pp. 542–552, Springer Berlin, 2006.
8. I. Haritaoglu, D. Harwood, and L. S. Davis, "W4: A real time system for detecting and tracking people," in *International Conference on Face and Gesture Recognition*, 1998.
9. P. L. Rosin and T. Ellis, "Image difference threshold strategies and shadow detection," in *The Sixth British Machine Vision Conference*, *Proc. BMVC*, pp. 347–356, 1995.
10. R. Cucchiara, M. Piccardi, and A. Prati, "Detecting moving objects, ghosts and shadows in video stream," *Pattern Analysis and Machine Intelligence* **25**, pp. 1337–1342, October 2003.
11. A. Prati, I. Mikic, M. M. Trivedi, and R. Cucchiara, "Detecting moving shadows: Algorithm and evaluation," *Pattern Analysis and Machine Intelligence* **25**, pp. 918–923, July 2003.
12. R. Cucchiara, C. Grana, M. Piccardi, A. Prati, and S. Sirotti, "Improving shadow suppression in moving object detection with HSV color information," in *Intelligent Transportation Systems*, *Proc. IEEE*, pp. 334–339, August 2001.
13. R. M. Haralick and L. G. Shapiro, *Computer and Robot Vision*, Addison-Wesley, Boston, 1992.
14. Q. Wan and Y. Wang, "Multiple moving objects tracking under complex scenes," in *The Sixth World Congress on Intelligent Control and Automation*, *Proc. IEEE* **2**, pp. 9871–9875, 2006.