An Efficient Background Updating Scheme for Real-time Traffic Monitoring

Suchendra M. Bhandarkar and Xingzhi Luo
Department of Computer Science, The University of Georgia
Athens, Georgia 30602-7404, USA

Abstract

Background updating is an important problem in dynamic scene analysis. It is critical to be able to distinguish between long-term gradual changes in the background and short-term rapid changes resulting from moving objects in the scene. In this paper we propose an efficient background updating scheme for real-time traffic monitoring. In particular, we address the *sleeping person* problem, which arises frequently in the context of real-time traffic monitoring. The proposed scheme combines two levels of reasoning: low-level reasoning based on pixel status analysis and high-level reasoning based on moving object correspondence analysis. The proposed scheme is robust and fast enough to satisfy the real-time constraints of traffic monitoring.

1. Introduction

Separating the foreground from background is an important though difficult problem in computer vision. This problem is even more complex in the case of dynamic scenes where the background typically changes with time and hence needs to be updated periodically. In computer analysis of dynamic scenes it is critical to be able to distinguish between long-term gradual changes in the background which are typically global in nature (such as changes in ambient illumination) and short-term rapid changes in the scene resulting from the presence of moving objects. Most object tracking systems need a background image to extract moving objects in the scene. Systems that use known background images for training [1] are not adaptive to changes in the background if the training images do not span all possible variations in the background. Moreover, in many practical situations, it is difficult to acquire training images that do not contain a moving object.

Some object tracking systems use adaptive techniques to update the background image on the fly such as by periodically computing the temporal average of the image frames [2]. A major shortcoming of the temporal averaging scheme is its inability to address the *sleeping person* problem [8]. The *sleeping person* problem arises frequently in the context of automated traffic monitoring when a moving vehicle stops in the scene (such as at a traffic light) and, on account of being motionless, is improperly merged with the background image [8]. Another shortcoming of temporal averaging is the presence of shadows in the background image especially in areas containing high/frequent motion. Replacing temporal averaging by median filtering addresses the shadowing problem but not the *sleeping person* problem [4].

Koller *et al.* [3] propose the following background updating scheme as an improvement over temporal averaging and median filtering:

$$B(x,y;t+1) = B(x,y;t) + \alpha(F(x,y;t) - B(x,y;t)) \quad (1)$$

where $B(x,y;t)$ is background image at time t, $\alpha = \alpha_1(1 - M(x,y;t)) + \alpha_2 M(x,y;t)$, $F(x,y;t)$ is the image frame at time t, $M(x,y;t)$ is the motion hypothesis mask given by

$$M(x,y;t) = \begin{cases} 1 & \text{if } |F(x,y;t) - F(x,y;t-1)| > T_t \\ 0 & \text{otherwise} \end{cases} \quad (2)$$

and $1 \gg \alpha_1 \gg \alpha_2 > 0$. This technique ensures that there are no shadows at the busy pixels in the background image, where moving objects pass frequently. However, it still cannot solve the *sleeping person* problem. In [5] and [6] it is assumed that reliable background pixels are those which do not exhibit motion for a long period of time. This assumption, however, is not valid if moving objects become static and remain static for a long time.

A more sophisticated background updating scheme proposed in [7] uses coarse object segmentation at the image block level to construct a block similarity matrix using motion information. This method can handle situations where a moving object becomes static for a given time interval. However, the size of the similarity matrix scales quadratically with the length of the time interval under consideration. The memory requirement and computational overhead make this technique unsuitable for real-time object tracking. Since the time interval for analysis is constrained by the limitations of memory and processing speed, if a moving object remains static for longer than this time interval, it is improperly merged with the background image. Techniques based on Kalman filtering [9] and linear prediction [8] have been proposed for background updating resulting in update equations similar to equation (1). Edge data is typically used to

track objects and these techniques work well in the absence of occlusion. However, in traffic monitoring, occlusion is a common occurrence thus limiting the applicability of these techniques. Occlusion occurs when portions of several objects in the 3D scene project onto a common region in the 2D image plane. Thus, only the object closest to the camera is visible in that region of the image plane.

The previously mentioned techniques perform poorly when confronted with the *sleeping person* problem [8]. We classify the *sleeping person* problem into two categories: the *middle-static-object* problem where a moving object becomes static in the middle of the frame sequence and the *initially-static-object* problem where an object is static at the beginning of the frame sequence and eventually moves. Neither the *middle-static-object* problem nor the *initially-static-object* problem are handled adequately by the previously described methods.

In this paper we propose a background updating scheme for real-time traffic monitoring that addresses the *sleeping person* problem by combining low-level reasoning based on the analysis of the status of individual pixels with high-level reasoning based on correspondence analysis of the moving objects in the scene. The high-level reasoning about inter-frame object correspondence enables robust background updating and detection of the *initially-static-object*. The low-level reasoning enables one to update the status of each pixel (indicating whether it belongs to the background or not) and also detect certain abnormal situations caused by an *initially-static-object*.

In the proposed scheme, the background updating is done on a frame-to-frame basis without requiring much history information to be stored. After the *initially-static-object* problem is solved, the background updating can be performed within a local window instead of the entire image. This reduces the processing overhead for background updating leaving more CPU time for object tracking and other tasks such as object recognition. By periodically moving the window around the image frame, the background can be refreshed to adapt to changes in illumination.

Most of the previously cited techniques detect moving objects by computing some measure of the inter-frame difference. The proposed technique, however, uses the background image to detect moving objects while simultaneously using the knowledge of the moving objects to update and detect errors in the background image. Thus, the processes of moving object detection and background updating are closely integrated. Experimental results show that it is possible to use the background image, before it is fully initialized, to detect a *middle-static-object*. The integration of moving object detection and background updating makes the system robust to large variations in the speeds of the moving objects within the field of view.

2 Background Updating Scheme

The proposed background updating scheme can be summarized as consisting of two primary steps: (a) Perform image segmentation on the first frame to initialize the background image (Section 2.1), and (b) Update the background using correspondence analysis and reasoning (Section 2.2).

The background image is maintained in an image buffer $B(x, y; t)$. Each pixel in $B(x, y; t)$ has the following attributes: *status*, which indicates whether the background pixel intensity is valid or not (1 represents valid, 0 represents invalid); *sum*, which represents the accumulation of gray levels at this pixel location since the last time when it was valid; *count* which is the total number of updates to the pixel since the last time when it was valid in the background image buffer; and g, which is the background gray level of this pixel given by $g = sum/count$. The temporal averaging used to compute the value of g produces a robust background, which adapts to changes in ambient illumination with the passage of time.

For each pixel, we design a set of operators to update the background image. The operator *Add* adds the gray scale value of the current frame $F(x, y; t)$ to $B(x, y; t)$ using the following rules:

1. If $B(x, y; t)$ is valid, that is $status = 1$, then

$$\begin{array}{lcl} sum & = & sum + F(x, y; t) \\ count & = & count + 1 \\ g & = & sum/count \end{array} \tag{3}$$

2. If the status of $B(x, y; t)$ is invalid, that is $status = 0$, then

$$\begin{array}{lcl} status & = & 1 \\ sum & = & F(x, y; t) \\ count & = & 1 \\ g & = & F(x, y; t) \end{array} \tag{4}$$

Equation (3) is used to update a background pixel which is already valid whereas equation (4) is used to initialize a background pixel which is still invalid. The operator *Invalidate* simply invalidates the status of the background pixel, i.e., $status = 0$. Initially, the status of all the background pixels is set to *invalid*. We also maintain a timer variable for each pixel, which accounts for the time that this pixel has been continuously contained within moving objects. If the pixel is not contained within any moving object at any time, we set the timer to 0. If the pixel is contained within moving objects continuously for too long, we *invalidate* the background at that pixel. This operator, termed as *Timer_Invalidate*, is disabled when the background is fully initialized and is stable.

The three operators: *Add*, *Invalidate* and *Timer_Invalidate* are the basic low-level pixel-based

reasoning operators. The first two are used to refresh the background image as described in Section 2.3. The third operator is used to solve the *initially-static-object* problem at the pixel level. When correspondence analysis and reasoning (Section 2.2) fail to detect the *initially-static-object* (which is initially misclassified as valid background) which then moves away, the object extraction method in Section 2.1 detects a false object at the initial location of the *initially-static-object* in every background frame thereafter. The timers associated with the pixels at the initial location of the *initially-static-object* count until they reach a given threshold value after which the *Timer_Invalidate* operator *invalidates* the *status* of these background pixels. If there is no moving object at these pixel locations after the *Timer_Invalidate* operation, the false object will no longer be detected and these background pixels are refreshed to the real background value using the first rule described in Section 2.3.

2.1 Background Image Initialization

We initialize the background image using inter-frame motion detection. We define the difference image as:

$$D(x, y; t) = |F(x, y; t) - F(x, y; t-1)| \qquad (5)$$

We use double thresholding to segment the moving objects in $D(x, y; t)$. The first threshold τ_1 is used to extract the core regions of the moving objects. The second threshold $\tau_2 < \tau_1$ is used to grow the core regions using spatial connectivity resulting in a binary image $T(x, y; t)$. Connected component labeling (CCL) and size filtering are used to identify significant connected regions in $T(x, y; t)$ which are then presumed to represent the moving objects in the scene. A bounding box is computed for each moving object.

We initialize the background image as follows: for every pixel (x, y) in the current frame $F(x, y; t)$, if (x, y) is not in the bounding box of any moving object, then we *Add* $F(x, y; t)$ to $B(x, y; t)$, else, do nothing (i.e., keep $B(x, y; t)$ invalid). The bounding box is used to reduce the effect of the object shadow.

The background updating process is prone to error if there exists a static object in the scene at the beginning of the frame sequence. Since this object cannot be extracted using motion detection, its pixels are misclassified as valid background pixels, causing problems in future updates to the background image. As described in the following subsections, the proposed scheme solves this problem using two levels of reasoning, one based on object correspondence analysis and the other based on a timer associated with each pixel location in the background image $B(x, y; t)$.

With the initialized background, we are able to refine the object extraction method. Instead of using equation (5) to get the difference image, we use the following equation:

$$\begin{aligned} D(x, y; t) \quad = \quad & \max(|F(x, y; t) - F(x, y; t-1)|, \\ & |F(x, y; t) - B(x, y; t) \cdot g| \\ & \times B(x, y; t) \cdot status) \end{aligned} \qquad (6)$$

in which, $B(x, y; t) \cdot g$ and $B(x, y; t) \cdot status$ are the attributes of $B(x, y; t)$ mentioned previously. Essentially, if the background pixel (x, y) is valid, we include it to extract the object, if not, ignore it. For a moving object that is going to stop, the value of $|F(x, y; t) - F(x, y; t-1)|$ exhibits a decreasing trend. But the value of $|F(x, y; t) - B(x, y; t) \cdot g| \times B(x, y; t) \cdot status$ can still be large if $status$ is 1. Thus, the *middle-static-object* problem is automatically prevented, since these pixels will not be misclassified as belonging to the background on account of the high value of $D(x, y; t)$ at these pixel locations. If there is no *initially-static-object* in the frame sequence, then a simplified version of our background updating scheme is as follows: (1) Compute the difference image using equation (6), (2) Extract moving object(s) using double thresholding and CCL, and, (3) Update the pixel locations where there are no moving objects present. We revert to this simplified scheme to refresh the background during object tracking when the *initially-static-object* problem has been solved and all the background pixels have been validated.

2.2 Correspondence Analysis and Reasoning

The correspondence analysis and reasoning phase detects the *initially-static-object* and refreshes the background image at the same time. During this phase, we also solve the problems created by the presence of object shadows and uniformity in object color. When the value of *count* is not large enough, the presence of object shadows will cause the background pixel intensity values to deviate significantly. It is important to minimize the effect of object shadows in the initial stages when the background image is not stabilized. Also, if an object with uniform color or intensity such as a car or truck moves very slowly and if the background status is invalid for the pixels contained within this object, then the motion information in the interior pixels of the object may be difficult to obtain. The background refreshing process itself makes the background image adaptive to the changes in illumination.

We use a simple scheme to predict the object speed for correspondence analysis instead of Kalman filtering [3] or linear prediction. We use an object's speed computed in a previous frame as it's predicted speed in the current frame. The result of the correspondence analysis is used to compute the actual speed of the object in the current frame and to update the prediction for the current frame. This method

works very well for traffic monitoring, where a moving object moves coherently. Correspondence analysis is observed to improve the overall robustness of background updating process. Moreover, errors in correspondence analysis are observed not to cause significant deviations in the gray level values of the background pixels.

Suppose the moving objects at time t and time $t-1$ are denoted by $O(t) = \{o_1^t, o_2^t, ..., o_n^t\}$ and $O(t-1) = \{o_1^{t-1}, o_2^{t-1}, ..., o_m^{t-1}\}$, respectively. For each object o_i^{t-1} we predict its new position ω_i^{t-1} using the predicted speed. We compute the overlapping area between ω_i^{t-1} and every object in $O(t)$ and construct the correspondence table C, where entry $C[i,j]$ denotes the size of the overlapping region between object ω_i^{t-1} and o_j^t.

We perform inter-object correspondence analysis based on the following categorization:

1. **One-to-One Correspondence:** Object $o_i^{t-1} \in O(t-1)$, corresponds to only one object $o_j^t \in O(t)$ and vice versa. Thus row i and column j in C are all 0's except for cell $C[i,j]$. Hence objects $o_i^{t-1} \in O(t-1)$ and $o_j^t \in O(t)$ are deemed to be the same physical object in successive frames.

2. **One-to-None Correspondence:** Object $o_i^{t-1} \in O(t-1)$ does not correspond to any object in $O(t)$.

3. **None-to-One Correspondence:** For an object $o_j^t \in O(t)$, there is no corresponding object in $O(t-1)$.

4. **One-to-Many Correspondence:** For an object o_i^{t-1} in $O(t-1)$, there is more than one corresponding object in $O(t)$, however, object o_i^{t-1} is the only corresponding object for each of them.

5. **Many-to-One Correspondence:** For an object o_j^t there is more than one corresponding object in $O(t-1)$, however, o_j^t is the only corresponding object for each of them.

6. **Many-to-Many Correspondence:** Objects o_i^{t-1} and $o_{i'}^{t-1}$ in $O(t-1)$ correspond to the same object in $o_j^t \in O(t)$. However, $o_{i'}^{t-1}$ also corresponds to another object $o_{j'}^t \in O(t)$. In this case, we use the following steps to simplify the correspondence, so that the result falls in one of categories 1–5 described above.

   (a) If object $o_i^{t-1} \in O(t-1)$ corresponds to several objects in $O(t)$, but one of the these objects, say o_j^t, corresponds to several objects in $O(t-1)$ (including o_i^{t-1}), then we examine column j in C, and set the smallest non-zero correspondence value in this column to 0.

   (b) If object $o_j^t \in O(t)$ corresponds to several objects in $O(t-1)$, but one of these objects, say o_i^{t-1}, corresponds to several objects in $O(t)$ (including o_j^t), then we examine the corresponding row i in C and set the smallest non-zero correspondence value in this row to 0.

Steps (a) and (b) are performed repeatedly until the resulting correspondence falls in one of categories 1-5 above.

2.3 Background Updating with Correspondence Reasoning

The results of the correspondence analysis are used to update the background image B as follows:

1. If a pixel (x,y) is not contained in any moving object in either $O(t-1)$ or $O(t)$, then we just *Add* the pixel value $F(x,y;t)$ to the background pixel $B(x,y;t)$ using the *Add* operator described previously.

2. In the case of *one-to-one* correspondence, we consider the object to be moving normally. A common occurrence is when an object which is static at the beginning of the frame sequence begins to move, it starts with a small size and then grows rapidly. Thus, when we detect an object whose area growth rate is larger than a threshold, we *invalidate* the background at the pixels within the bounding box of that object. This solves some instances of the *initially-static-object* problem. Otherwise, if a point (x,y) is contained in the bounding box of the object at time $t-1$ but not in the bounding box of the corresponding object at time t, then we *Add* the pixel value in the current frame to the background as in case 1 above.

3. In the case of *many-to-one* correspondence, we use a bounding box which is the smallest one that includes the many bounding boxes and then handle the relation as if it were a *one-to-one* correspondence. The reason we use the merged bounding box is to be able to handle the situation where an object has uniform color. In such a case, it is difficult to estimate the motion in the interior pixels of the object using temporal differencing. Consequently, the extracted object is often fragmented into more than one component. It is possible for pixels lying in areas between these components (which truly belong to the object) to be misclassified as background pixels. This results in false updates to the background which could cause the background image quality to deteriorate, especially when it has not been completely initialized.

4. In the case of *one-to-many* correspondence, we use the same reasoning as in the case of *many-to-one* correspondence.

5. In the case of *one-to-none* correspondence, we have an object moving out of the scene or just random noise. We simply *Add* the pixels in the bounding box to the background after a chosen time delay, such as 100 frames. The time delay is chosen to minimize the effect of random motion in the image (due to noise) on the background update.

6. In the case of *none-to-one* correspondence, we have an object that has stayed in the background since the beginning of the frame sequence and has now begun to move or just random noise. In this case, we just invalidate the background at pixels within the bounding box of the object. This solves most instances of the *initially-static-object* problem.

After the background is initialized, we can use a simple version of background updating that can run in real time. We use equation (6), double thresholding and CCL to extract the moving object(s). However, we do not perform correspondence analysis. We only update the background pixels, where there is no object present, using equation (3).

The computational requirements of the background updating procedure can be scaled down, if required, to meet the temporal constraints of real-time traffic monitoring. The background updating procedure can be performed only within a window of predetermined size and position in the background image. Thus, the object extraction and correspondence analysis, which are used to update background, are performed only within the window, where the window size is chosen to be larger than the largest object in the scene. The window is then moved periodically within the image frame in a prespecified manner in order to update the entire background image. The moving window-based approach can be used to update the background image after it is initialized so that computational resources can be made available for other tasks such as object recognition and object tracking.

3 Experimental Results

Our experiments on traffic monitoring videos show very good results when there is no static object at the beginning of the frame sequence. When a static object moves into the field of view and becomes motionless, it is never merged into the background image. When there is an *initially-static-object*, the correspondence reasoning sometimes fails to detect that it has begun to move because of the presence of an occluding object. As described in Section 2, the *Timer_Invalidate* operator can solve this problem eventually if the traffic is not very heavy. When the *initially-static-object* moves out of its position, a false object will be detected at that position in the background image. The timer associated with the pixels at this position will count until the elapsed time exceeds a threshold, after which these pixels will perform a *Timer_Invalidate* operation. If there is no moving object at that position at this time, the real background value will be copied into these pixels with the *Add* operator in the next frame. We also notice that when there is heavy traffic, it takes a long time for the pixel values in the background image to converge to their true background values, after the *Timer_Invalidate* operation has been performed. This is because in heavy traffic it is possible to encounter another static object at the very pixel position that is being invalidated, thus preventing the true background value from being copied into that pixel via the *Add* operation.

Figure 1 shows a series of snapshots of the background updating process. The initial background image (Figure 1(b)) is just a copy of the second frame (Figure 1(a)). However, the pixel status is invalid in the area where motion is detected. The white car in the center of the field of view is static in the beginning and hence appears in the background image (Figure 1(b)). Even when the white car moves away in frame 1328 (Figure 1(c)), it persists in the background image due to the presence of other vehicles that partially occlude it (Figure 1(d)). However, by frame 6700 (Figure 1(e)), the timer mechanism ensures that all pixels are updated to their true background values (Figure 1(e)). The result of the background updating scheme based on simple temporal averaging of the frames is shown in Figure 1(g). The background image can be observed to contain the stationary vehicles waiting at the traffic light (i.e., the *sleeping person* problem). Koller's scheme (equation (1)) also has a serious problem when dealing with static objects, since it can adapt very quickly to static objects in the scene causing them to merge with the background image.

Figure 2 shows a comparison of the converge rates of the proposed scheme, Koller's scheme and temporal averaging scheme for background updating. The y axis in Figure 2 represents the percentage of pixels that do not match the real background values within a prespecified threshold. The x-axis represents time measured in terms of the number of frames (at a uniform sampling rate of 30 frames/second). The proposed background updating scheme can be seen to converge more rapidly to an overall background value that is much closer to the real background value compared to the other two schemes.

4 Conclusions

In this paper we proposed a background updating scheme for a real-time traffic monitoring system. For the *sleeping person* problem, we considered two cases termed as the *initially-static-object* problem and the *middle-static-object* problem. The former was handled with the pro-

(a) Image Frame 2

(b) Background Frame 2

(c) Image Frame 1328

(d) Background Frame 1328

(e) Image Frame 6700

(f) Background Frame 6700

(g) Background Frame 6700 with temporal averaging

Figure 1: Snapshots of the background updating procedure

posed two-level reasoning scheme. The two-level reasoning scheme was shown to also handle the problem of uniform color object and shadows. The high-level reasoning based on inter-object correspondence solved the *initially-static-object* problem quickly when there was little occlusion. In the event that the high-level reasoning failed to solve this problem (in cases of heavy occlusion), the low-level reasoning based on the *Timer_Invalidate* operator solved the problem (even in the presence of heavy occlusion) though it took longer. The *Timer_Invalidate* also handled the *middle-static-object* problem. In summary, the proposed background updating scheme was seen to be robust and scalable with respect to the number of moving objects in the scene and changes in background illumination. It did not entail training on typical background images, had low computational complexity and was fast enough to satisfy the temporal constraints of real-time traffic monitoring.

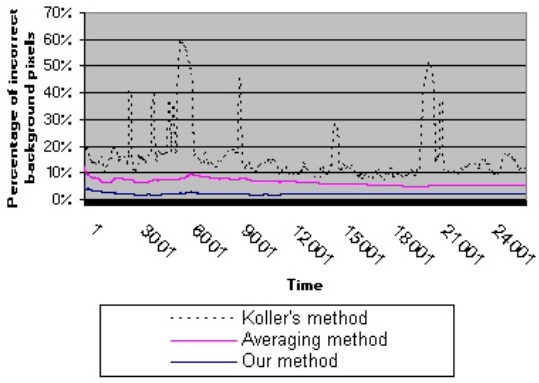

Figure 2: Convergence curve

References

[1] M. Isard and J. MacCormick, BraMBLe: A Bayesian Multiple-Blob Tracker, *Proc. Intl. Conf. Computer Vision*, Vol.2, Vancouver, Canada, July 2001, pp. 34-41.

[2] S. Kamijo, Traffic Monitoring and Accident Detection at Intersections, *IEEE Trans. Intelligent Transportation Systems*, Vol. 1. No. 2, June 2000, pp. 108-118.

[3] D. Koller, J.W. Weber and J. Malik. Robust Multiple Car Tracking with Occlusion Reasoning, *Proc. European Conf. on Computer Vision*, Stockholm, Sweden, May 1994, pp. 189-196.

[4] M. Massey and W. Bender, Salient stills: Process and practice, *IBM Systems Journal*, Vol. 35, Nos. 3&4, 1996, pp. 557-573.

[5] S.Y. Chien, S.Y. Ma, and L.G. Chen, Efficient Moving Object Segmentation Algorithm Using Background Registration Technique, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 12, No. 7, July 2002, pp. 577-586.

[6] T. Meier and K. N. Ngan, Automatic Segmentation of Moving Objects for Video Object Plane Generation, *IEEE Trans. Circuits and Systems for Video Technology*, Vol. 8, No. 5, Sept. 1998, pp. 525- 538.

[7] D. Farin, P.H.N. de With, and W. Effelsberg, Robust Background Estimation for Complex Video Sequences, *Proc. IEEE Intl. Conf. Image Processing*, Barcelona, Spain, Sept. 2003, pp. 145-148.

[8] K. Tooyama, J. Krumm, B. Brumit and B. Meyers, Wallflower: Principles and Practice of Background Maintenance, *Proc. Intl. Conf. Computer Vision*, Corfu, Greece, Sept. 1999, pp. 255-261.

[9] C. Ridder, O. Munkelt, and H. Kirchner, Adaptive background estimation and foreground detection using Kalman-filtering, *Proc. Intl. Conf. Recent Adv. Mechatronics*, Istanbul,Turkey, August 1995, pp. 193-199.