REAL-TIME ADAPTIVE BACKGROUND SEGMENTATION

Darren Butler, Sridha Sridharan

Queensland University of Technology
GPO Box 2434
Brisbane QLD 4001, Australia
{de.butler, s.sridharan}@qut.edu.au

V. Michael Bove, Jr.

MIT Media Laboratory
20 Ames St
Cambridge MA 02139, USA
vmb@media.mit.edu

ABSTRACT

Automatic analysis of digital video scenes often requires the segmentation of moving objects from the background. Historically, algorithms developed for this purpose have been restricted to small frame sizes, low frame rates or offline processing. The simplest approach involves subtracting the current frame from the known background. However, as the background is unknown, the key is how to learn and model it. This paper proposes a new algorithm that represents each pixel in the frame by a group of clusters. The clusters are ordered according the likelihood that they model the background and are adapted to deal with background and lighting variations. Incoming pixels are matched against the corresponding cluster group and are classified according to whether the matching cluster is considered part of the background. The algorithm has been subjectively evaluated against three other techniques. It demonstrated equal or better segmentation than the other techniques and proved capable of processing 320×240 video at 28 fps, excluding post-processing.

1. INTRODUCTION

Video cameras are ubiquitous. They are essential for industries such as surveillance, communications and entertainment resulting in an incalculable amount of video being captured, processed and stored every day. Technological advances have enabled the development of systems that analyse digital video scenes automatically. Fundamental to many of these systems is the need to segment moving objects from the background. Historically, computational complexity has restricted automatic background segmentation to small frame sizes, low frame rates or offline processing.

As *a priori* knowledge of a scene's background does not often exist, the key for any background segmentation algorithm is how to learn and model it. The simplest approach is to calculate an average background frame whilst no moving objects are present. Subsequently, when an object enters the scene, it will cause the current frame to diverge from the average background frame and its presence can be easily detected by thresholding the difference between the frames. However, any physical change or illumination change of the scene's background will severely degrade the algorithm's performance indefinitely. To counteract this problem, the average background image must be continuously adapted to incorporate the background variations.

The assumption that the background is strictly stationary is also problematic. Consider a tree branch waving in the wind. It moves and yet should still be incorporated into the background model. We define a pseudo-stationary background as one in which all constituent objects are motionless or undergo small repetitive motions. Clearly, even small motions can cause large scale variations in the observed pixel values. Therefore, the adaptive background difference algorithm, as outlined, is insufficient for pseudo-stationary backgrounds.

As early as 1988, the usefulness of segmenting moving objects from a static background was recognised. Even given the limited hardware of the time, Seed and Houghton presented an analysis of a number of algorithms for the maintenance of a background frame in real-time under varying ambient conditions [1]. However, the problem domain was constrained to segmenting vehicles from a roadway. Of the techniques, the two most promising were *Random updating* and *Slope limited updating*. Random updating replaces background pixels by the corresponding pixels in the current frame according to a pseudo-random sequence. As no reference is made to what data the pixels actually contain, errors in the background frame will occur. However, the errors are isolated and can be reduced by subsequent processing. Conversely, Slope limited updated places restrictions on the amount background frame pixels must differ from their counterparts in the current frame before they will be updated and by how much they can be changed.

Chien *et al* improve the basic background differencing algorithm by progressively learning the background frame [2]. They surmise that the longer a pixel remains roughly the same, the more probable that it belongs to the background. Pixels are classified as stationary by thresholding the difference between consecutive frames. A count of the number of frames that a pixel has remained stationary is maintained and when the count is sufficiently high, the pixel is copied to the background frame. Whilst this technique succeeds in learning and adapting the background frame, it fails to handle pseudo-stationary backgrounds.

Stauffer and Grimson recognised that pseudo-stationary backgrounds are inherently multi-modal and hence modelled each pixel in the frame with a mixture of gaussians [3]. Incoming pixels are compared against the corresponding gaussian mixture model (GMM) and a match is sought. A match is defined as a pixel value within 2.5 standard deviations of a gaussian. If a match is found, the parameters of the matching gaussian are adjusted accordingly. However, if no match can be found, the least probable distribution is replaced with a distribution that models the incoming pixel. Gaussians that are more frequently matched are more likely to model background pixels and so incoming pixels are classified accordingly. This algorithm has since been enhanced to improve its learning rate and to account for object shadows [4].

A less obvious advantage of Stauffer and Grimson's algorithm is its ability to adapt rapidly to transient background changes. For instance, if an object enters the scene and then stops moving it will eventually be incorporated into the background model. If it then moves again, the system should rapidly recognise the original background as corresponding gaussians may yet remain in the mixture model. However, maintaining a GMM for every pixel is an enormous computational burden and results in low frames rates when compared to the algorithm of Chien *et al*. This paper proposes a segmentation algorithm with a similar premise to that of Stauffer and Grimson but having the capability of processing 320×240 video in real-time on modest hardware.

2. ALGORITHM

The premise of our algorithm is the more often a pixel takes a particular colour, the more likely that it belongs to the background. Therefore, we require a technique for maintaining information regarding the history of pixel values. We model each pixel by a group of K clusters where each cluster consists of a weight w_k and an average pixel value or centroid c_k. Additionally, the algorithm assumes that the background region is stationary. That is, the camera geometry must be fixed or there must be compensation for background changes due to camera motion.

Incoming pixels are compared against the corresponding cluster group. The matching cluster with the highest weight is sought and so the clusters are compared in order of decreasing weight. A matching cluster is defined to have a Manhattan distance (i.e. sum of absolute differences) between its centroid and the incoming pixel below a user prescribed threshold T. If no matching cluster is found, the cluster with the minimum weight is replaced by a new cluster having the incoming pixel as its centroid and a low initial weight.

If a matching cluster was found then the weights of all clusters in the group are adjusted according to:

$$w_k = w_k + \frac{1}{L}\left(M_k - w_k\right) \tag{1}$$

where M_k is 1 for the matching cluster and is 0 for the remaining clusters. The parameter L is simply the inverse of the traditional learning rate, α. It controls how quickly scene changes are incorporated into the background model. Smaller values for L result in faster adaptation and larger values result in slower adaptation.

The centroid of the matching cluster must also be adjusted according to the incoming pixel. Previous approaches adjust the centroid based on a fraction of the difference between the centroid and the incoming pixel. However doing so results in fractional centroids and inefficient implementations. We chose instead to accumulate the error between the incoming pixel and the centroid. When the error term exceeds $L - 1$ the centroid is incremented and when it is below $-L$ it is decremented. This is approximately equivalent to adjusting the centroid on every frame using $c_k = c_k + \frac{1}{L}\left(x_t - c_k\right)$ but avoids the need for fractional centroids and can be implemented very efficiently as is detailed in Section 3.

After adaptation, the weights of all clusters in the group are normalised so that they sum up to one using:

$$w_k = \frac{w_k}{S} \;\; ; \forall k \qquad \text{where } S = \sum_k w_k \tag{2}$$

The weights necessarily total to one and are treated somewhat like probabilities because they represent the proportion of the background accounted for by the cluster.

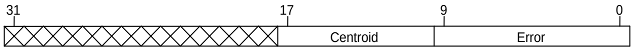

Fig. 1. Centroid bit assignment: $[B = 9 \Leftrightarrow L = 512]$.

The normalised clusters are next sorted in order of decreasing weight. However, note must be taken of the new location of the matching cluster. Maintaining the clusters in order of weight aids both the initial cluster comparisons and the final classification step. Pixels are classified by summing the weights of all clusters that are weighted higher than the matched cluster. That is:

$$P = \sum_{k > M_k}^{K} w_k \tag{3}$$

The result, P is the total proportion of the background accounted for by the higher weighted clusters and is an estimate of the probability of the incoming pixel belonging to the foreground. Larger values of P are evidence the pixel belongs to the foreground and smaller values are evidence that it belongs to the background. This value can be thresholded to obtain a binary decision or can be scaled to produce a grayscale alpha map.

3. IMPLEMENTATION

In implementing the algorithm of Section 2 we specifically targeted Y'CbCr 4:2:2 video. Hence, it was necessary to make minor modifications to the basic algorithm. Clusters are formed with both a luma centroid and a chroma centroid. The luma centroid consists of two adjacent luma components (Y_1', Y_2') and the chroma centroid holds the corresponding chroma components (Cb, Cr). The use of two centroids is beneficial as it allows the specification of separate thresholds for luma and chroma. Given only a single threshold it is likely that luma would dominate the matching process. With this modification, a match is defined to occur only when the Manhattan distance for both the luma and chroma components are below their respective thresholds.

The update equations for the cluster weights are unchanged but we employ a clever trick to efficiently accumulate the error terms and update the centroids. As aforementioned, the error term lies in the range $[-L, L-1]$. We can equivalently shift this range to $[0, 2L-1]$ by the addition of L. That is, we simply shift the origin of the error term from 0 to L. If we now restrict L to be of the form $L = 2^B$ then the entire range of the error term can be specified in $B + 1$ bits. Furthermore, adaptation of the centroid now occurs when accumulation of the error term results in overflow or underflow of the $B + 1$ bits.

Only 8 bits are required to specify any of the components of Y'CbCr video. Therefore, if we represent the components of the centroids (Y_1', Y_2', Cb, Cr) by 32 bit integers then the remaining 24 bits can be used to accumulate the error term. This gives a maximum range for B of $[0, 23]$ or equivalently, $L \in \{1, 2, 4, 8, ..., 8388608\}$. The user specifies the number of bits B used to hold L and we shift upwards the components of the centroids by $B + 1$ bits accordingly. The lower bits are used to accumulate the error term which is initialised to the origin L (see Figure 1). By virtue of this formulation, when overflow or underflow of the error term occurs, the centroid is neatly adjusted automatically.

Careful consideration of the cluster weight update equation (Eq.1) reveals another suboptimality of the basic algorithm. It is not difficult to show that if the weights of a cluster group sum up to one, after application of the update equation they will still sum up to one. Therefore, it is sufficient to only normalise the weights (Eq.2) when a matching cluster could not be found forcing the creation of a new cluster.

Another property of Equation 1 is that the weights of the unmatched clusters are downscaled by the same factor (i.e. $w_k = \frac{L-1}{L} \times w_k$). Consequently, only matching clusters and newly created clusters have the potential to be unsorted. Furthermore, as the weight of a matched cluster increases, it is only necessary to sort in the direction of the higher weighted clusters.

4. POST PROCESSING

There are two kinds of misclassifications that may occur in segmentation results. False positives occur when background regions are incorrectly labeled as foreground. Conversely, false negatives occur when foreground regions are classified as background. Post processing aims to reduce the number of such misclassifications without an appreciable degradation in classfication speed.

False positives resemble pepper noise and are typically attributed to camera noise. That is, they are small (1-2 pixel), incorrectly classified regions surrounded by correctly classified background pixels. The conventional morphological open operation can be used to reduce false positives whilst preserving the contours of correctly classified regions.

False negatives arise because of the existance of similarities between the colours of foreground objects and the background. They form holes in correctly classified foreground regions and can be quite large. Consequently, they are more difficult to remove than false positives. Typically, a connected components algorithm is used to find connected foreground classifications. Next, small regions, which are presumed to be false positives, are eliminated. Finally, the holes in the remaining regions are filled.

To reduce the misclassification rate of our implementation, we began by joining pixels separated by a single pixel gap. Next we extracted the contours of all regions that were classified as foreground using features of the Open CV library[1]. If the area enclosed by the contour was below a threshold it was eliminated. Any remaining contours were presumed to correspond to real foreground objects and were retained. This technique proved sufficient to eliminate most of the false positives and false negatives culminating in results like those of Section 5.

5. RESULTS

We have qualitatively compared our algorithm against three other background segmentation methods. The first method, *VAR*, calculates an average frame and a variance frame from a buffer of past frames. Frames are segmented based upon the variance normalised difference between them and the average frame. The algorithm, *GMM1*, is according to the original work of Stauffer and Grimson [3], whereas, *GMM2* includes the modifications made by KaewTraKulPong and Bowden *et al* [4]. Finally, *NHD* corresponds to the algorithm that is described in this paper. The results were obtained under Linux using a dual 1000-Mhz Pentium III computer with 512 Mb of RAM. However, no attempts were made to parallelise the code and only a single processor was utilised. Table 1 summarises the run-time performance of the four algorithms without post-processing and Figure 2 compares their final segmentations after post-processing.

Table 1. Algorithm features & performance.

Algorithm	**Video Format**	**User Params.**	**Frame Rate (fps)**	**Mem. Usage (Mb)**	**CPU Usage (%)**
VAR	Y'CbCr	8	29	11.9	25
GMM1 [3]	RGB	2	9	23.9	97
GMM2 [4]	RGB	2	13, 6[1]	25.4	97
NHD	Y'CbCr	4	28	13.0	92

[1] The initial transient frame rate of the GMM2 algorithm and its steady-state frame rate respectively.

A key issue for segmentation algorithms is their computational complexity. Table 1 clearly demonstrates that our algorithm is about four times more computationally complex than the simple variance based segmenter. However, it is still capable of processing 28 frames per second and as shown in Figure 2 achieves significantly better segmentation. Furthermore, our algorithm adapts the background model on every frame whereas the variance based segmenter only adapts once every three hundred frames. Increasing the adaptation rate results in an increased CPU utilisation.

The post-processing stage used to generate the results was identical for all four algorithms. Although the technique performs well in general, it is certainly not optimal for each segmenter. Therefore, the results should not be construed as the best segmentation obtainable from each algorithm. However, the results do demonstrate that our algorithm performs as well or better than other recently published techniques whilst achieving significantly higher frame rates.

6. CONCLUSIONS

The need for fast and accurate algorithms for segmenting moving objects from the background is undeniable. In this paper, we have presented a new background segmentation algorithm targeting real-time applications. The algorithm models each pixel in the frame with a group of K clusters and adapts the clusters to deal with variations in both the background and the ambient lighting. Incoming pixels are compared against the corresponding cluster group using the Manhattan distance and are classified accordingly.

Currently, our algorithm does not differentiate between moving objects and their shadows. Consequently, shadows often cause the segmentation to blur or they are detected as separate moving objects. In either case, the effect is undesirable and future work aims to reduce the impact of shadows without significantly degrading the achieved frame rate.

Finally, we qualitatively evaluated our algorithm against three other techniques. Our algorithm demonstrated equal or better segmentation than the other techniques and has proven capable of processing 320×240 video at 28 frames per second, excluding post-processing. Therefore, as desired, the algorithm is suitable for deployment in real-time applications.

[1] The Open Source Computer Vision Library is used courtesy of the Intel Corporation and is available for public download from the World Wide Web at "http://www.intel.com/research/mrl/research/opencv/".

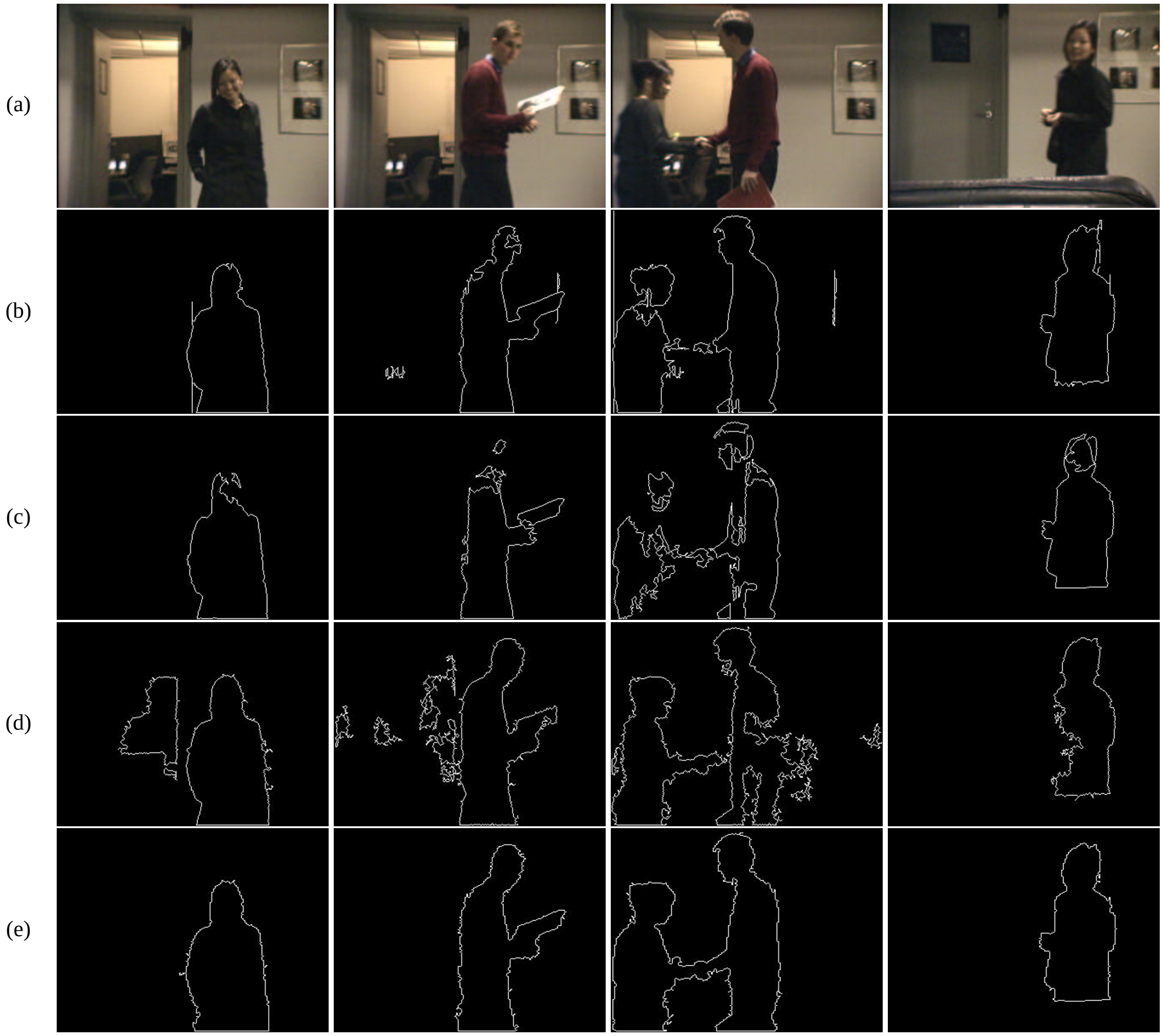

Fig. 2. Segmentation results: (a) Original, (b) VAR, (c) GMM1, (d) GMM2, (e) NHD.

7. REFERENCES

[1] N.L. Seed and A.D. Houghton, "Background updating for real-time image processing at TV rates," in *SPIE Vol. 901, Image Processing, Analysis, Measurement and Quality*, 1988, pp. 73–81.

[2] Shao-Yi Chien, Shyh-Yih Ma, and Liang-Gee Chen, "Efficient moving object segmentation algorithm using background registration technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, pp. 577–586, Jul 2002.

[3] Chris Stauffer and W.E.L Grimson, "Adaptive background mixture models for real-time tracking," in *Proceedings of CVPR '99*, Jun 1999, pp. 246–252.

[4] P. KaewTraKulPong and R. Bowden, "An improved adaptive background mixture model for real-time tracking with shadow detection," in *Proceedings of AVBS01*, Sep 2001.