MVA2007 IAPR Conference on Machine Vision Applications, May 16-18, 2007, Tokyo, JAPAN

Robust Real Time Multi-Layer Foreground Segmentation

Simon Denman, Vinod Chandran, Sridha Sridharan
Image and Video Research Laboratory
Queensland University of Technology
GPO Box 2434, Brisbane 4001, Australia
{s.denman,v.chandran,s.sridharan}@qut.edu.au

Abstract

Many surveillance applications (object tracking, abandoned object detection) rely on detecting changes in a scene. Foreground segmentation is an effective way to extract the foreground from the scene, but these techniques cannot discriminate between objects that have temporarily stopped and those that are moving. We propose a series of modifications to an existing foreground segmentation system [1] so that the foreground is further segmented into two or more layers. This yields an active layer of objects currently in motion and a passive layer of objects that have temporarily ceased motion which can itself be decomposed into multiple static layers. We also propose a variable threshold to cope with variable illumination, a feedback mechanism that allows an external process (i.e. surveillance system) to alter the motion detectors state, and a lighting compensation process and a shadow detector to reduce errors caused by lighting inconsistencies. The technique is demonstrated using outdoor surveillance footage, and is shown to be able to effectively deal with real world lighting conditions and overlapping objects.

1 Introduction

Foreground segmentation techniques are used to separate the foreground objects from a known or learned background. This is commonly used as a first step in many computer vision applications such as surveillance systems, and as such is often followed by additional processes to further segment the image. Stauffer and Grimson [5] proposed an algorithm where each pixel was modeled by a GMM, where incoming pixels are compared to the GMM to determine how well they match the background. Harville et al. [3] proposed allowing a higher level process to impose positive or negative feedback to force changes in the background model as needed; and [6], proposed the addition of shadow removal and a foreground support map to aid in the updating of the background. However, modeling each pixel with a GMM is very processor intensive, and not ideal when foreground segmentation is only the first step in a multi-step process (i.e. surveillance). To address this, Butler et al. [1] proposed using an approximation to a GMM, where each pixel is modeled as a group of clusters (a cluster consists of a centroid, describing the pixels colour; and a weight, denoting the frequency of its occurrence).

Whilst these techniques are able to separate foreground from background; they cannot segment overlapping foreground objects. Kim et al. [4] proposed a system to address this using a codebook to model individual pixels. Statistics for each possible code are recorded to determine which codes belong to background and foreground, and which belong to short-term background (i.e. stopped cars).

We propose an adaptation to [1] that incorporates the motion history of objects and performs multi-layer foreground segmentation. The proposed method distinguishes between not only static (background) and dynamic (foreground) regions, but also other categories that may be temporarily static but not part of the background. We also describe a feedback mechanism that allows the background model to be altered by an external process, so that objects of interest (i.e. abandoned objects) can be held out of the background whilst other changes can still be incorporated. In addition we propose adding a variable threshold, shadow detection and lighting compensation to improve performance.

We demonstrate the improved foreground segmentation on outdoor surveillance footage, and show that its capable of accurately distinguishing between active foreground objects, static foreground objects and a static background; and handling shadows and illumination changes typical of outdoor surveillance footage.

2 Segmentation Algorithm

2.1 Existing Technique

An efficient method of foreground segmentation that is robust and adapts to lighting changes was proposed by Butler [1] based on modeling of pixel attributes in multi-modal distributions and pixel clustering. The technique was extended by [2] to incorporate optical flow and improve performance. In this work, a one-frame history of each pixel was stored in the form of an index of the matching cluster for each pixel. The method is further extended into a multi-layer framework here using such motion information.

Let $f(x, y, t)$ be a frame sequence, where x, y is in $[0, N-1]$ and t is in $[0, T]$. Let $P(x, y, t')$ be a pixel in the frame at time t'. Pixels are tracked with their motion and colour history over time interval δt, and have data stored in a set of K

clusters, $C(x,y,t,1..K) = (y_1, y_2, Cb, Cr, w)$, which represent a multi-modal PDF. Input images are in Y'CbCr 4:2:2 format. Pixels are paired to create a cluster which consists of two luminance values (y_1 and y_2), a blue chrominance value (Cb), and red chrominance value (Cr) to describe the colour; and a weight, w. Clusters are ordered from highest to lowest weight; and the current matching cluster, $C(x,y,t,m)$ (where m is the index of the matching cluster in the range $1..K$), for each pixel is stored, giving an approximation of the image.

For each (x,y,t) the algorithm makes a decision assigning it to one of the sets (background, or a motion layer) by matching $P(x,y,t)$ to $C(x,y,t,k)$, where k is an index in the range 1 to K. Clusters are matched to incoming pixels by finding the highest weighted cluster which satisfies

$$|P(y_1) - C(k)(y_1)| + |P(y_2) - C(k)(y_2)| < LumThr \quad (1)$$

$$|P(Cb) - C(k)(Cb)| + |P(Cr) - C(k)(Cr)| < ChrThr \quad (2)$$

where $P = P(x,y,t)$ and $C(k) = C(x,y,t,k)$. The centroid of the matching cluster is adjusted to reflect the current pixel colour, and the weights of all clusters in the pixels group are adjusted to reflect the new state.

$$w_k = w_k + \frac{1}{L}(M_k - w_k) \quad (3)$$

where w_k is the weight of the being adjusted; L is the inverse of the traditional learning rate, α; and M_k is 1 for the matching cluster and 0 for all others. If there is no match, then the lowest weighted cluster is replaced with a new cluster representing the incoming pixels. Clusters are gradually adjusted, allowing the system to adapt to changes in the background.

Based on the accumulated pixel information, the frame can be classified into foreground;

$$fgnd = \forall(x,y,t)\ where \sum_{i=0}^{m} C(x,y,t,i)(w) < T(x,y,t) \quad (4)$$

where $T(x,y,t)$ is the foreground/background threshold; and background. The foreground can be further split into moving and temporarily static objects. We define a static region as an area of motion that has entered the scene and stopped moving, and an active region as an area of motion that is currently moving. The separation of these regions is explained in section 2.2.

2.2 Static Layers

To discriminate between active and static foreground, we need to compare against the last cluster at a given pixel, and any static foreground objects that are present there.

When $C(x,y,t,m) = C(x,y,t-1,m)$, $P(x,y,t)$ has a static layer, $S(z)$, initialised, where z is the depth of the layer. Each layer has a counter, c, and a colour, (y_1, y_2, Cb, Cr) associated with it. For subsequent frames where $C(x,y,t,m) = C(x,y,t-1,m)$, $P(x,y,t).S(z).c$ is incremented, otherwise it is decremented. Static pixels can be defined as

$$\forall(x,y,t) \in fgnd\ where\ P(x,y,t).S(z).c >= \delta t \quad (5)$$

Static pixels can be further organised into layers depending on when the pixel appears. Layers can be built one on top of the other, as new objects appear and come to a stop atop an existing static layers. Layers remain until the observed cluster is matched to either a lower layer, or the background.

The number of static layers available, L_s, is determined by the parameters of the background model and the requirements of the scene. One cluster must be dedicated to the active foreground and there must be one cluster per background mode. Given this, the maximum number of static layers is

$$L_s = K - L_b - 1 \quad (6)$$

where K is the total number of clusters in the background model and L_b is the number of background modes. Typically, we set $L_s = 2$ and $K = 6$.

The algorithm for detecting and updating the static layers for a single pixel is outlined in figure 1.

Each static layer is monitored by a counter which is updated each time step, and used to determine the state of the layer (i.e. static, to be removed). Counters are incremented when the layer is detected, and decremented only when a lower level static layer (or background) is detected. When a higher level static layer (or active layer) is detected counters are unchanged as the static layer may be hidden below. Counters are decremented gradually to provide error tolerance for incorrect cluster matching, or noise. The decrement rate depends on the scene, with more challenging scenes requiring a slower decrement rate due to the increased chance of an erroneous cluster match. Layers are removed when the counter reaches zero, and counters are capped to guarantee that a layer can be removed in a set number of frames.

The algorithm has some limitations in that it can only detect overlaps when at least one of the overlapping objects is static. It is also not possible to determine when a lower level static object leaves while higher level static objects remains, or when a lower level objects moves in behind a higher level object, due to the relevant pixels being obscured.

2.3 Variable Threshold

A variable threshold is added to the motion detection to aid the system in handling different lighting conditions within the same scene (i.e. shadow, sunlight, artificial light). The threshold, $T(x,y,t)$, at $P(x,y,t)$ is dependent on the weight of the highest weighted (most commonly occurring) cluster, $C(x,y,t,K)$. A higher weight indicates a more consistent and stable background, allowing a tighter threshold to be applied.

$$T(x,y,t) = T_{max} - (C(x,y,t,K).w \times (T_{max} - T_{min})) \quad (7)$$

where T_{max} is the maximum threshold and T_{min} is the minimum threshold. This process is applied to both the luminance and chrominance thresholds.

The cluster learning rate is such that lower weighted clusters increase in weight faster than higher weighted clusters, so if $T(x,y,t)$ becomes too low and motion is incorrectly detected, $C(x,y,t,K).w$ will be reduced substantially by a match to a lower weighed cluster, increasing $T(x,y,t)$ and

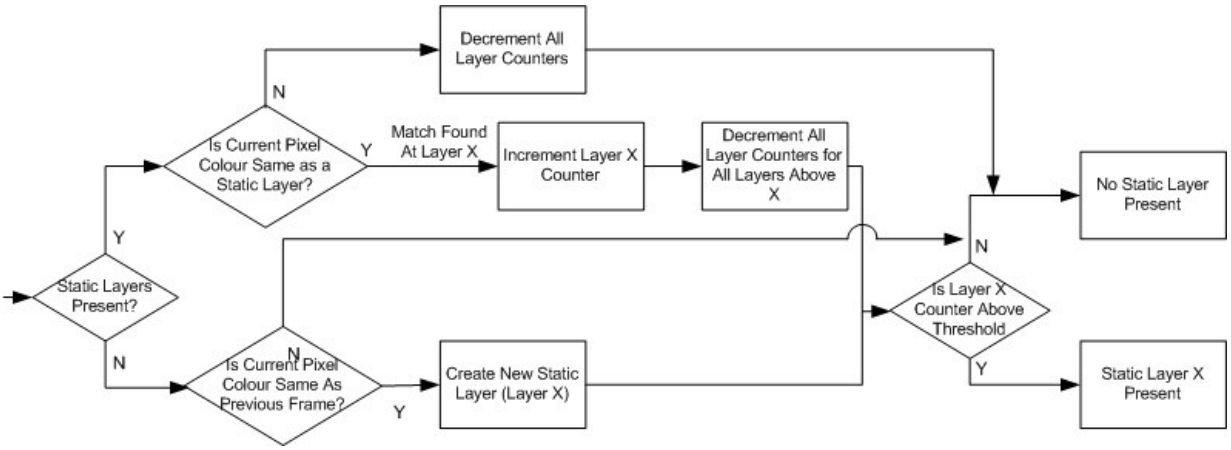


Figure 1. Static Layer Matching Flowchart - If the pixel already has static layers, we compare against these. If there are no layers, or no matches to existing layers, we check to see if there is possibly a new static layer forming (last two frames have the same colour at the pixel).

returning $P(x, y, t)$ to a state of no motion. This results in the thresholds for each pixel being able to reach, and approximately remain at, a natural equilibrium.

2.4 Feedback

It is important to allow changes to occur in the background model as the scene varies, but we must also prevent foreground objects of interest being incorporated into the background. As it is not practice for the motion detector to make these decisions, we propose a method where by an external process can make these decisions.

The inverse of the weight adjustment algorithm can be used to prevent the object from being incorporated into the background model, by effectively stopping all weight updates so that objects of interest remain in the foreground.

$$w_k = \frac{(Lw_k - M_k)}{L - 1} \tag{8}$$

where w_k is the weight of the cluster being adjusted; L is the inverse of the learning rate (lower values will result in background changes being incorporated faster); and M_k is 1 for the matching cluster and 0 for all others.

2.5 Lighting Compensation

In surveillance situations, lighting levels can change rapidly resulting in large amounts of erroneous motion. To prevent this we propose incorporating simple adjustment to the luminance threshold to compensate for lighting changes.

For each frame, the total luminance (Lum_f) is calculated by summing the luminance values for all pixels that are detected as background, and the highest weighted background cluster ($C(x, y, t, K)$) for pixels detected in motion. We omit the motion pixels to ensure that a strongly coloured foreground object does not have an adverse effect on the system, and that we are only considering changes that are caused by the lighting. Lum_f is used to update the background luminance (Lum_b, the total luminance in the background model), which is subject to the same learning rate as the background model.

$$Lum_b(t) = Lum_b(t-1)\frac{L-1}{L} + Lum_f\frac{1}{L} \tag{9}$$

For each frame the ratio between Lum_b and Lum_f is calculated, and used to calculate the offset for the luminance threshold ($LumAdj$). This is added to the luminance threshold when processing the next frame, so that the system will be tolerant of the lighting change, but remain sensitive to other motion.

$$LumAdj = \left(\frac{max(Lum_b, Lum_f)}{min(Lum_b, Lum_f)} - 1\right) * 255 \tag{10}$$

To cope with lighting changes in isolated areas of the image, we subdivide the image into a grid, and calculate the luminance adjustment for each subregion. The appropriate threshold is then applied when processing each region.

2.6 Shadow Detection

Shadows can result in motion being detected where there is none. As such, it is important to recognise shadows and ensure that they are not recorded as motion. Shadows can be characterised by the fact that they alter the luminance component of the objects colour, but have minimal effect on the chrominance. We add shadow detection to the algorithm by adding additional constraints when matching the incoming pixels to the clusters.

$$0 < (C(k)(y_1) - P(y_1)) + (C(k)(y_2) - P(y_2)) < ShadThr \tag{11}$$

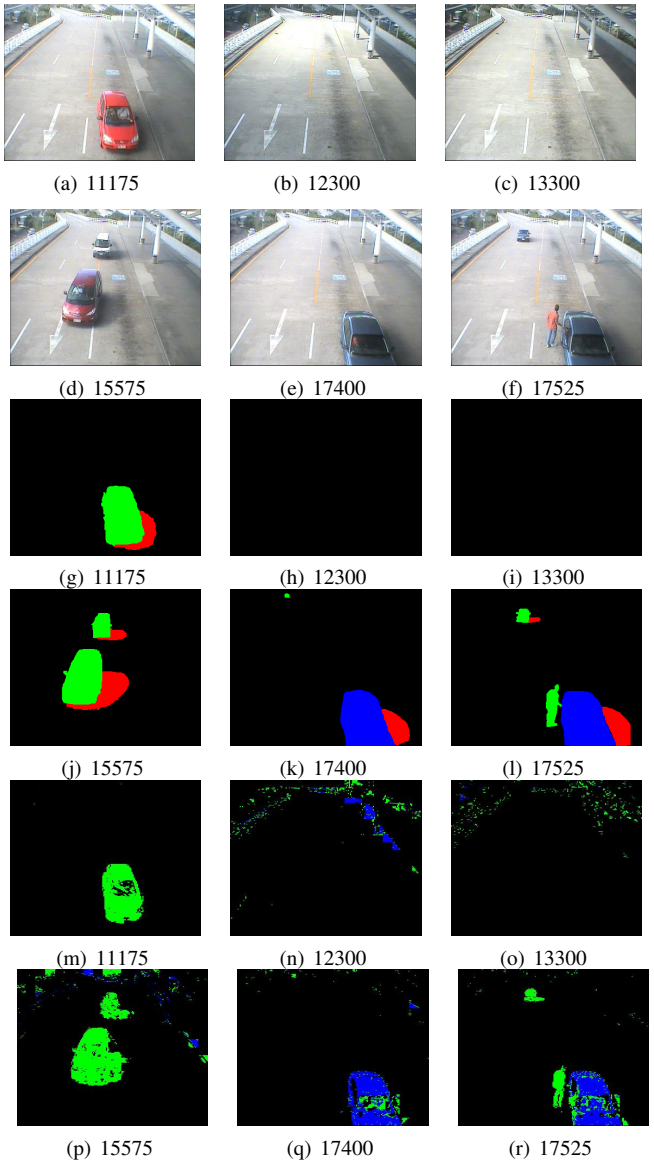
$$|P(Cb) - C(k)(Cb)| + |P(Cr) - C(k)(Cr)| < (ChrThr/S) \tag{12}$$

If there is a positive difference in the luminance, less than the prescribed shadow threshold, $ShadThr$, and only a small difference in the chrominance (determined by dividing the chrominance threshold, $ChrThr$, by an integer S) we have a shadow and motion is not detected at P.

3 Results

Testing was conducted using a 10000 frame sequence of real world data acquired at a public passenger drop off area. Ten frames which illustrated various effects such as lighting variation, shadows and overlapping objects were hand segmented for comparison (it is not practice to hand segment the entire sequence). Performance was measured in terms of false negatives (FN, motion present in ground truth but

not detected) and false positives (FP, motion detected but not present in ground truth). The algorithms overall performance was compared to Butler's [1] (see table 1). Incorrect detection of the motion type results in a FN and a FP being recorded for the appropriate motion types (i.e. active foreground detected when static's expected - FN for static, FP for active; static detected in layer two expected in layer one - FN and FP for static). We measure the performance of the algorithm at classifying active motion, static motion and shadows, to provide an indication of the performance of each component. A simple object detector was applied to the output of our algorithm to locate foreground objects and apply feedback to the region they occupy. No morphological operations were applied to the output of either system.



(a) 11175 (b) 12300 (c) 13300

(d) 15575 (e) 17400 (f) 17525

(g) 11175 (h) 12300 (i) 13300

(j) 15575 (k) 17400 (l) 17525

(m) 11175 (n) 12300 (o) 13300

(p) 15575 (q) 17400 (r) 17525

Figure 2. Multi-layer segmentation results - (a)-(f) original images; (g)-(l) ground truth; (m)-(r) motion detection output; green indicates active motion, blue static motion, red in the ground truth images indicates shadow (which we expect to be detected as no motion in the bottom row). The captions for each image indicate the frame number.

Table 1. Motion Detection Results

	Our Algorithm		Butler's Algorithm [1]	
	FN	FP	FN	FP
Active Motion	1.35%	21.60%	N/A	N/A
Shadow Motion	N/A	26.82%	N/A	58.00%
Static Motion	0.33%	40.85%	N/A	N/A
Total Motion	2.19%	32.24%	17.83%	53.94%

As table 1 and figure 2 show the system performs well and is able to discern between static and active foreground objects, as well as cope with lighting changes (see frames 12300 and 13300 in figure 2) and shadows. However, the system does struggle to deal with lighting various where the background is widely varied, due to the different textures in the region (i.e. the area around the rails on the left edge of the image, see frame 13300 and 15575). The shadow detection can also effect the motion detection when dark objects enter, such as the windscreen and windows of the car in frames 17400 and 17525. Despite the limitations of proposed changes however, they result in a significant improvement in performance, significantly reducing the rate of false positives and false negatives when compared to [1]. The modified algorithm is capable of running in real time. For the test sequence, our algorithm achieved an average execution time per frame of 28.6 ms running on a 3.0GHz Pentium 4 processor.

4 Conclusion

We have described a modified foreground segmentation process that allows multiple layers of foreground to be detected, and utilises an variable threshold, shadow detection and luminance compensation to assist in handling varied light conditions. We have also described a feedback process that allows the background model to be adjusted by an external source, providing a means to keep an object from becoming part of the background. Future work will focus on further improving the lighting compensation, adding adaptability to the shadow detection and increasing the speed of the algorithm.

References

[1] D. Butler, S. Sridharan, and V. M. Bove Jr. Real-time adaptive background segmentation. In *ICASSP*, 2003.

[2] S. Denman, V. Chandran, and S. Sridharan. Adaptive optical flow for person tracking. In *DICTA*, Cairns, Australia, 2005.

[3] M. Harville. A framework for high-level feedback to adaptive, per-pixel, mixture-of-gaussian background models. In *ECCV*, volume 3, Copenhagen, Denmark, 2002.

[4] K. Kim, D. Harwood, and L. S. Davis. Background updating for visual surveillance. In *ISVC*, pages 337–346, 2005.

[5] C. Stauffer and W. Grimson. Adaptive background mixture models for real-time tracking. In *CVPR*, volume 2, page 252 Vol. 2, 1999.

[6] H. Wang and D. Suter. A re-evaluation of mixture-of-gaussian background modeling. In *ICASSP*, pages 1017–1020, Philadelphia, USA, 2005.