A Spatially Distributed Model for Foreground Segmentation

Patrick Dickinson, Andrew Hunter, Kofi Appiah
Center for Visual Surveillance and Machine Perception, University of Lincoln, Lincoln, UK

Abstract

Foreground segmentation is a fundamental first processing stage for vision systems which monitor real-world activity. In this paper we consider the problem of achieving robust segmentation in scenes where the appearance of the background varies unpredictably over time. Variations may be caused by processes such as moving water, or foliage moved by wind, and typically degrade the performance of standard per-pixel background models.

Our proposed approach addresses this problem by modeling homogeneous regions of scene pixels as an adaptive mixture of Gaussians in color and space. Model components are used to represent both the scene background and moving foreground objects. Newly observed pixel values are probabilistically classified, such that the spatial variance of the model components supports correct classification even when the background appearance is significantly distorted. We evaluate our method over several challenging video sequences, and compare our results with both per-pixel and Markov Random Field based models. Our results show the effectiveness of our approach in reducing incorrect classifications.

Key words: Foreground segmentation, Background model, Spatial coherence, Mixture of Gaussians

1. Introduction

The purpose of an automated visual surveillance system is to extract meaningful information from an image sequence. A series of "bottom-up" processing steps is typically applied to each new image frame, each eliciting a more refined and descriptive representation of the observed scene.

The lowest level of processing is applied to the entire set of image pixels. The aim is to identify regions of interest (usually moving objects) for further processing. Without making prior assumptions about appearance, an effective approach is to first build a model of the empty scene, or background. New foreground objects may then be segmented by comparison: "background subtraction" removes those pixels which closely match the background model, leaving a residual subset of pixels corresponding to foreground objects. Typically, further stages of processing cluster the foreground pixels into object representations, track objects from frame-to-frame, and infer relevant behavioral characteristics. The process of background subtraction is critical to the system performance, as segmentation errors reduce the effectiveness of subsequent processing. However, this remains a challenging task outside of laboratory conditions.

Until recently, most background subtraction schemes have employed a per-pixel background model. Previous observations are used to construct a background representation for each pixel location: this may include intensity, color, and possibly other features. The background subtraction process then independently classifies each pixel: the new observed value is compared with its model, and labeled as either foreground or background. There are many examples of per-pixel models. Haritaoglu's "W4" [4] system models each pixel's background as a mean value, with lower and upper ranges of tolerance. A statistical model is used by Wren's "Pfinder" system [28], which maintains adaptive Gaussian distributions. The model parameters are estimated from previous observed values, and new observations with low probability are labeled as foreground. The Wallflower system [25] uses a Weiner filter to predict pixel background values from a linear combination of recent observations.

We can expect that the appearance of the scene background will change over time; for example, due to gradual changes in lighting, or small movements by background objects. To account for this, systems such as those described above employ adaptive models. However, some changed pixels will be misclassified during the adaptation process. Moreover, simple adaptation is not enough to capture more complex background processes. Dynamic backgrounds, which exhibit repeating spatio-temporal variations, are

common: for example, outdoor scenes often include foliage which may move in the wind.

In such cases, background values observed at a single pixel may be generated by more than one process. Per-pixel models have therefore been developed which represent multi-modal behavior. For example, Elgammal [2] uses a non-parametric kernel density estimate to model background distributions. The most widely adopted model is that proposed by Stauffer and Grimson [23].

Stauffer uses an adaptive mixture of Gaussians (MoG) to model observations of each pixel's process in RGB color space. Thus, at time t, the probability of observing a new color value $\mathbf{x}_{i,t}$ at pixel i is given by:

$$p\left(\mathbf{x}_{(i,t)}|\Theta_{(i,t)}\right) = \sum_{k=1}^{K} \omega_{(i,t)}^{k} \eta\left(\mu_{(i,t)}^{k}, \Sigma_{(i,t)}^{k}\right) \quad (1)$$

Where $\Theta_{(i,t)} = \left\{\theta_{(i,t)}^{1} \ldots \theta_{(i,t)}^{K}\right\}$ are the mixture model parameters estimated at time t, K is the (fixed) number of components, η is the multivariate Gaussian probability distribution function, and

$$\sum_{k=1}^{K} \omega_{(i,t)}^{k} = 1 \quad (2)$$

Typically, between 3 and 5 components are used, and highly weighted components are taken to be generated by background processes. Each new pixel value is matched against existing components. If it is matched to a background component, the pixel is labeled as background; or foreground otherwise. The model is then updated to incorporate the new observation.

Stauffer's multi-modal scheme allows a time-varying background to be modeled on a per-pixel basis, provided that the model is suitably parameterized, and that each pixel's background modes are frequently presented. This model has become a de facto standard in automated surveillance, and much research has been directed at refining it. For example, KaewTraKulPong [8] proposes a different model update procedure, and a normalized color space; Harville [5] adds depth information from a stereo camera, and sets a separate learning rate for each pixel; Shimada [21] and Cheng [1] have both recently investigated optimal model orders; and Tian [24] has developed modifications to deal with illumination changes, and shadowing.

Despite its popularity, there are a number of well documented limitations to the per-pixel MoG model. Variations which are sporadic, or where one mode dominates, are still not well represented. Unfortunately such variations are common where the underlying process is erratic, for example, moving foliage. Similarly, as Tian [24] notes, foreground objects are absorbed at different rates at different pixels, causing object fragmentation. Fragmentation problems also arise where foreground objects overlap spatially with background objects of similar color.

These types of errors are systemic under the assumption of an independent pixel model. Scene images are generated by a set of discrete objects (both background and foreground) such that pixel values generated by the same object exhibit a strong spatial, chromatic, and temporal coherence. Such relationships are not represented by a per-pixel model, but can be used to address the above classification problems, and to produce a more robust segmentation in general.

The work we present here is directed at this goal. We begin by reviewing some existing approaches to this problem, and proceed to describe our algorithm in detail. We conclude with an experimental comparison of our algorithm with Stauffer's original, and also with a recent Markov Random Field (MRF) based model, in sequences with challenging spatio-temporal backgrounds variations.

2. Previous Work

Interest has grown recently in background models which represent spatial relationships between pixels, and in this section we review some existing work. In particular we examine methods which, like ours, explicitly model spatial distributions in the background. We also pay particular attention to Random Field based methods, which we use for comparison in our evaluation.

Although per-pixel models do not directly express spatial relationships between pixels, some authors have modified the classification process to account for perturbations in scene structure. In this type of approach, classification incorporates not only the background model of the pixel, but also the models of pixels in its local neighborhood. Each pixel model is still independent, and so no account is made of the overall coherency of segmentation; however, misclassifications may be locally eliminated.

At the simplest level, Elgammal [2] adds a neighborhood comparison to his non-parametric model. Pixels which do not match their own background model are compared to the models in a small neighborhood. A similar, but more comprehensive, approach is taken by Ren's "Spatially Distributed Gaussians" (SDG) [16] to account for global background transformations caused by a moving camera. This system first registers global image features to estimate a global translation between the background model and observed image. This estimate is used as the starting point for a search for matches between an observed pixel value and a neighborhood of background models. A MoG background is used, and a background label is generated for any match in the search area.

Spagnolo [22] takes a slightly different approach, incorporating neighboring values directly into the matching value of each pixel. A radiometric similarity measure is used to compare an observed pixel value with its background value, and incorporates values from the local neighbourhood. Classification is performed by thresholding the similarity. Pixel-wise temporal difference is first generated from the two most recent frames, and used as a mask for the background subtraction.

Toyama [25] also uses pixel neighborhood color values to identify pixels incorrectly labeled as background. Strongly

supported regions of foreground pixels are used to build color histograms, and then to seed areas for connected components expansion against their color model. Although this approach still employs a per-pixel model, there some consideration of how the structure of a segmented object can be used to support pixel classification.

3. Sub-space Background Representations

Eigenbackgrounds [12] represent the background as a set of dominant eigenvectors, extracted from a set of training images. This sub-space identifies regions of the image which are invariant, and foreground pixels are identified by comparing a new image with its projection through the sub-space back to image space. Monnet [11] develops this by employing an on-line auto-regressive model to predict changes in background structure. This method captures repeating variations in the spatial structure of the scene background, but is inflexible as only variations presented during training are represented.

An interesting and unusual approach is taken by Seki [19], who uses co-occurrence of adjacent spatial features to model the spatio-temporal structure of a time-varying background. A series of background training frames are divided into blocks of pixels, and eigen-decomposition [12] is used to represent each block in an appropriate sub-space. The training data is used to learn model temporal correlations between adjacent blocks, such that an observed value for a block may be used to predict the appearance of its neighbors. Sets of blocks which are not well correlated in an input image are considered to have low background likelihood. This offers an improvement over eigenbackgrounds, as the correlations are localized. However, as these spatial relationships are learned the model still lacks the flexibility to deal with unpredictable variations.

3.1. *Random Fields*

The methods discussed above are somewhat limited: they are either modifications of pixel-based methods, or learn patterns of invariance to detect unusual observations. A more flexible and useful model is one which can express general spatial properties of a scene's structure. For this reason, Markov Random Fields (MRFs), and (more generally) Conditional Random Fields (CRFs), have recently received some recent attention as a foreground segmentation method.

MRFs are probabilistic graphical models in which each node represents a random variable, and edges between nodes represent dependencies. In the case of foreground segmentation schemes, nodes represent pixel labelings $\{foreground, background\}$, and each node is connected to its spatial 4 or 8 pixel neighborhood. The edge dependencies express the Markovian nature of the local node dependency: given its neighbors, each node is conditionally independent of the rest of the field. Given an observed image $\mathcal{I}$, and suitable observation likelihood function, the aim is to estimate the labeling $\mathcal{L}$ which maximizes the posterior (MAP estimate):

$$p(\mathcal{L}|\mathcal{I}) = p(\mathcal{I}|\mathcal{L})\, p(\mathcal{L}) \tag{3}$$

This is made tractable by the Hammersley-Clifford theorem which by re-expresses the dependencies as a set of node-clique potentials. Quantifying these potentials defines the prior probability for a given global field labeling. Assuming an independent observation likelihood for each pixel, the probability of a single node label change can be estimated by simply summing energy terms. This forms the basis of standard MAP-MRF labeling techniques. The advantage of these techniques is that clique potentials which impose suitable spatial dependencies can be easily expressed at pixel level. An appropriate observation likelihood can also be defined for each pixel, and the MAP estimation results in a segmentation which is globally optimal.

Paragios [13] proposed a MRF segmentation scheme for a subway monitoring system which used normalized color as an observation likelihood. Pixel-wise gradient observations were also used to relax spatial continuity constraints across discontinuities. The iterated conditional modes algorithm is used to estimate the optimal labeling. More recently, Schindler [18] uses a per-pixel MoG model to develop the observation likelihood, and resolves the field using a graph cutting algorithm. Sheikh [20] also uses a graph cutting algorithm in conjunction with a kernel density estimate to build background and foreground observation likelihoods. Both these methods uses 4-neighborhood cliques to simplify the graph.

Wang has sought to incorporate temporal as well as spatial constraints by incorporating a Hidden Markov Model (HMM) into a MRF framework [26] and by using a CRF [27]. In both cases, a third label and corresponding observation likelihood is added to represent areas of shadow.

The MRF scheme proposed by Migdal and Grimson [10] develops directly from Stauffer's per-pixel MoG model. The MoG model is used to initialize the field for each frame, and the dominant background distribution is used as a background observation likelihood. Foreground likelihood is modeled as a uniform distribution in RGB color space. This model also includes temporal dependencies by linking each pixel to its previous labeling. We have chosen this algorithm as a benchmark for our work, and so we describe some implementation details here.

The Hammersley-Clifford theorem formulates the probability of a field labeling $\mathcal{L} \in \Phi$, where Φ is the set of all possible labellings, as a Gibbs distribution:

$$p(\mathcal{L}) = \frac{e^{-U(\mathcal{L})/T}}{Z} \tag{4}$$

Where Z is a normalisation constant, T is a temperature term used in the annealing process, and $(\mathcal{L})$ is an energy function such that:

$$U(\mathcal{L}) = \sum_{c \in C} V(\mathcal{L}) \quad (5)$$

Migdal uses Gibbs sampling to estimate the MAP field labeling, proposing a linear annealing schedule for T over a fixed number of field iterations. The clique potentials $V(\mathcal{L})$ define the spatial and temporal constraints, and are applied to pair-wise cliques in a pixel's 8-neighborhood. Values for the pair-wise clique potentials are not specified, but we have experimented with a range of values in our evaluation.

3.2. *Spatially Distributed Model Components*

A scene object is likely to generate pixels which are spatially coherent, and share similar color attributes. Consequently, a number of authors [14,6,7,3] have proposed background (and foreground) models in which clusters of homogeneous pixels are represented parametrically in an extended feature space. A typical feature spaces includes spatial and color information, such that a single pixel is represented by a 5-dimensional vector $\mathbf{x}_t = [x, y, R, G, B]^T$. Classifying an individual pixel involves assigning it to the model component most likely to have generated it.

Pixels are not then explicitly labeled as foreground or background: a model component may represent either part of the foreground or the background, so labeling is implied by association. Model adaptation may be implemented by re-estimating the components from their assigned pixels.

In a probabilistic framework this approach amounts to expressing learned spatial relationships in the observation likelihood, rather than, in the case of random fields, applying spatial dependencies as a prior. Like per-pixel models, spatial distributions may be learned and updated as the scene evolves, but considerably fewer components are required.

This type of representation naturally captures small changes in background structure. Small movements of background objects generate new background pixels which still have a high likelihood under the background model. In addition, there are a number of other advantages over the more typical processing architecture. Pixel clustering is more usually executed after classification, as a second foreground processing stage [9], [28]. Using clusters directly for segmentation integrates the two processes such that model adaptation automatically effects frame-to-frame object correspondence, and changes in scene structure are fed back to the next image classification step. It also allows foreground and background to be defined at the object level rather than pixel level: attributes such as size, or movement, can be used to specify which pixels are foreground and which are background.

A partial implementation of this type of model is presented by Raja's object tracker [15] which builds an off-line MoG model of color distributions for a known object and scene background. A spatial component is not included in the likelihood function, but implemented more simply as an axis-aligned bounding box approximating the extents of the object. Pixels inside the bounding box are classified as foreground or background according to their likelihood of their observed color value.

A more principled approach is taken by Pece [14], using pixel intensity rather than color. In this system, the spatial foreground components are represented by a Gaussian distribution, and the intensity as a uniform distribution. The background distribution is uniform in space, and exponential in pixel intensity. For each new image, each pixel is assigned to the most likely component, and the components are updated using Expectation Maximisation (EM). Foreground clusters are added and removed to adapt the model as objects appear and disappear from the scene.

Heisele [6] uses an iterative K-Means algorithm to cluster pixels in 5-Dimensional space. The model is adapted appropriately, and clusters with similar trajectories are grouped to form object hypotheses. Recently, Huac [7] has built on this work by explicitly classifying clusters as background or foreground. Spatial ambiguities are resolved by defining an elliptical search area for each cluster derived from the spatial covariance of its assigned pixels.

The distribution model used by Greenspan's video indexing system [3] bears some similarity to ours. Spatial and color cluster coordinates are modeled separately as independent Gaussian distributions, and time is added to give a 6-dimensional feature space. A video sequence is split into sub-sequences, and each is segmented separately. This is a two stage process: first, an appropriate model order is estimated, and then it is used to segment each image.

The modeling process using EM, and the Minimum Description Length principle to estimate the optimal model order. This involves building a series of models for each subsequence, and then selecting the most appropriate: consequently there is a considerable processing overhead. The sub-sequences are then aligned by building correspondences between components in successive sub-sequences, which allows objects to be tracked across the entire sequence.

Unlike our system, Greenspan's is unsuitable for on-line processing. Firstly, it is necessary to capture and sub-divide the sequence before segmentation can be applied. Secondly, model estimation requires that many models are built and compared for each sub-sequence, incurring a high processing overhead. Greenspan's system also exhibits model discontinuity at the transition from one sub-sequence to the next: a separate correspondence scheme is needed to track objects across transitions.

4. Our Approach

In our system we model homogeneous regions of the scene using an adaptive mixture of Gaussians in 5-dimensional feature space. Each pixel observation is represented by a corresponding feature vector $\mathbf{x}_t = [x, y, Y, U, V]^T$ where color is encoded using the YUV format. The probability distribution function for each model component is given by:

$$p(\mathbf{x}_t|\theta_{(j,t)}) = \omega_{(j,t)} \frac{e^{-\frac{1}{2}(\mathbf{x}_t-\mu_{(j,t)})(\Sigma_{(j,t)})^{-1}(\mathbf{x}_t-\mu_{(j,t)})}}{\sqrt{(2\pi)^d|\Sigma_{(j,t)}|}} \quad (6)$$

Where the parameters $\theta_{(j,t)} = \left\{\omega_{(j,t)}, \mu_{(j,t)}, \Sigma_{(j,t)}\right\}$ are the component weight, mean, and covariance matrix of the j^{th} component at time t, and the dimensionality, d, is 5. For k components, the general mixture model conditions given by equations (1) and (2) also hold in their appropriate form.

Given a new observed image, and a set of model parameters, an observed pixel value may be classified by assigning it to the component with the maximum posterior probability, C_{map}. Using log likelihoods:

$$C_{map} = argmax_j \left\{log(p(\mathbf{x}_t|\theta_{(j,t)})\right\} \quad (7)$$

We have simplified the model slightly by assuming that the spatial and color distributions are independent and uncorrelated. The distribution function in equation (7) is re-expressed as the product of a 2-dimensional spatial Gaussian and a 3-dimensional color Gaussian, with parameter sets $\theta^s_{(j,t)}$ and $\theta^c_{(j,t)}$. Each pixel value is then expressed by corresponding spatial vectors $\mathbf{x}^s_t = [x, y]^T$, and color vector $\mathbf{x}^c_t = [Y, U, V]^T$. Hence, equation (7) becomes:

$$C_{map} = argmax_j \left\{log(p(\mathbf{x}^s_t|\theta^s_{(j,t)}) + log(p(\mathbf{x}^c_t|\theta^c_{(j,t)})\right\} \quad (8)$$

4.1. *Implementation Overview*

We use model components to represent both background and foreground regions of the scene, under the premise that such a region is generated by a single corresponding process, such as part of an object. Background components are initialized from the first image of a sequence, in which it is assumed no foreground objects appear. Foreground components are introduced as required, in response to the appearance of pixel values which are not well represented by the background. Each component is explicitly labeled as $L_c \in \{foreground, background\}$, and pixels are implicitly labeled according to the component to which were assigned using equation (8).

The current assignments are stored in an image "support map". Figure 1 depicts a model instantiation corresponding to a frame from one of our test sequences. In this visualization the components are represented by rendering their mean color value at each pixel where they are spatially dominant.

All model components are updated by the statistics of their assigned pixels. Background components are updated more slowly than foreground components, reflecting the expectation that foreground will exhibit more dynamic behavior. The initialisation, assignments, and update procedures are described in more detail in the remainder of this section.

4.2. *Building the Background Model*

The initial set of background components are constructed from the first frame of the sequence. We have already described how Greenspan's system [3] uses EM to build a maximum likelihood parameter set for a similar Gaussian mixture. This technique is effective and well principled, but there are some problems using it for on-line processing. Firstly, a large number of iterations are required, making it computationally expensive. Secondly, we wish to adapt the model dynamically, in response to changes in scene structure. EM is proven to converge for a fixed data set, however, our data set changes with each new input image: thus, for example, we may need to re-estimate the model order when new objects enter the scene. Greenspan deals with this by dividing the video into closed sections, and building a separate model for each. However, this is not suitable for on-line processing.

The technique of splitting and merging components has been used by Raja [15] and by Pece [14] as a technique for dynamically adapting model order. We use an iterative splitting and merging technique to build an initial set of background components. We find that this method is computationally manageable, and, by minimizing the variance of components, generates an appropriate representation of the major regions of the scene. We use the following procedure to build the components:

1 The model is initialized with a single component, and each pixel's support map entry is set.
2 It is iteratively split until a suitable number of new components have been generated.
3 Pairs of similar components are then merged.
4 Any components which are spatially disconnected (possibly representing more than one object or process) are split.

4.3. *Splitting a Background Component*

A single iteration of the splitting procedure described in step 2 divides an existing component into two new ones. Given the background image, and current set of components, we first find the component with the highest spatial variance.

We calculate the principle eigenvalue, λ^s_j and corresponding eigenvector, Λ^s_j for each component's spatial covariance matrix. The component $C^s_{sp} = argmax_j \left\{\lambda^s_j\right\}$ is selected. Let $\mathcal{I}_{sp}$ be the pixels currently assigned to this component. If its eigenvalue $\lambda^s_{sp} > T^s_{sp}$, where T^s_{sp} is a predefined threshold, then it is split. We create a new component and re-assign to it those pixels $\mathbf{x} \in \mathcal{I}_{sp}$ which satisfy:

$$(\mathbf{x}^s - \mu^s_{sp}) \cdot \Lambda^s_{sp} > 0 \quad (9)$$

This amounts to placing a separating plane through the spatial mean, perpendicular to Λ^s_{sp}. The parameters of both components are then re-estimated from the statistics of their respective assigned pixels:

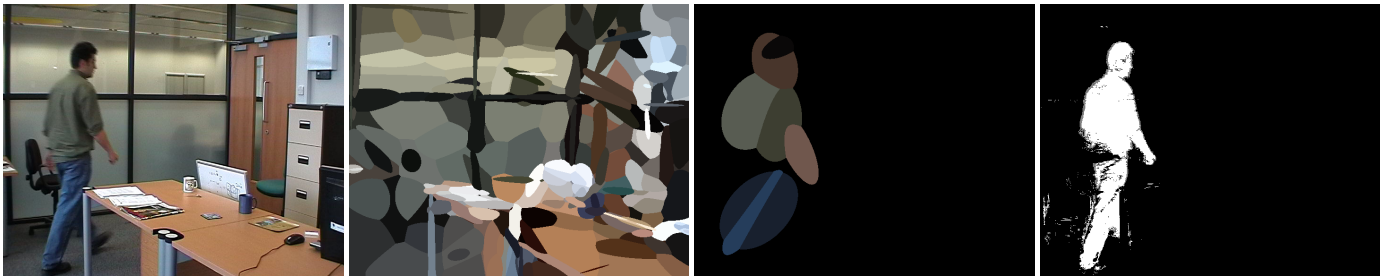

Fig. 1. Background and Foreground Models. Left to right : Original image, background model, foreground model, Foreground pixel set. In this visualization the model components are rendered with their mean color value, covering the area in which they are spatially dominant.

$$\omega_j = \frac{n_j}{N} \tag{10}$$

$$\mu_j = \frac{1}{n_j} \sum_{\mathbf{x} \in \mathcal{I}_{sp}} \mathbf{x} \tag{11}$$

$$\mathbf{z}_j = {\mu_j}^T \, \mu_j \tag{12}$$

$$\Sigma_j = \frac{\sum_{\mathbf{x} \in \mathcal{I}_{sp}} \mathbf{x}^T \mathbf{x}}{n_j} - \mathbf{z}_j \tag{13}$$

Where n_j is the number of pixels assigned to the component, and N is the total number of pixels in the image, and the value of μ_j used in equation (12) is the new value calculated using equation (11). We then apply the same selection and splitting procedure in color space, using a corresponding threshold T^c_{sp}, to split the component with the highest color variance.

We repeat this process, alternating between highest spatial and color variances, until reaching a maximum number of components, or until the largest found eigenvalues fall below their thresholds. The procedure is initialized with the single component built in step 1. The parameters of this component are estimated from the entire set of image pixels, using equations (10) to (13) with $n_j = N$.

4.4. *Completing the Initial Background Model*

We can now merge any similar components. If $\mathcal{M}^s_j(\mathbf{x}^s)$ is the spatial Mahalanobis distance of $\mathbf{x}^s$ from μ^s_j, and $\mathcal{M}^c_j(\mathbf{x}^c)$ is the color Mahalanobis distance of $\mathbf{x}^c$ from μ^c_j, then a pair of components is considered suitable for merging if the following holds:

$$\mathcal{M}^s_1(\mu^s_2) < T^s_{mg} \quad \wedge \quad \mathcal{M}^s_2(\mu^s_1) < T^s_{mg} \quad \wedge$$
$$\mathcal{M}^c_1(\mu^c_2) < T^c_{mg} \quad \wedge \quad \mathcal{M}^c_2(\mu^c_1) < T^c_{mg} \tag{14}$$

Where T^s_{mg} and T^c_{mg} are predefined thresholds. We consider each pair of components, and merge the qualifying pair with the lowest value of $max(\mathcal{M}^c_1(\mu^c_2),\ \mathcal{M}^c_2(\mu^c_1))$. This procedure is repeated until no qualifying pairs remain.

We next seek to identify components which represent spatially disconnected regions, and split them to represent those regions separately. The purpose of this step is to identify single components which represent more than one background process, and separate them. We order the components in descending value of λ^s_i, and step through the list. For each, we use a connected components algorithm to determine if it represents two or more disconnected regions of the support map: if so, we split the largest region away from the rest as a new component. For reasons of efficiency we implement this at a reduced resolution. We repeat this until no disconnected components are found, or for a maximum number of iterations. Finally, when this process is complete, components which have a zero or very small weight are culled from the model.

4.5. *Assigning Image Pixels to Model Components*

When a new image frame is captured, each pixel is assigned to its most likely model component, using equation (8). The pixel's support map entry is updated to record the assignment. The spatial variance Σ^s_i for large background components is typically high. This frequently results in pixels being assigned to regions from which they are significantly disconnected, and in particular, hampers detection of new foreground regions. To resolve this we apply the additional restriction that a pixel may only be assigned to a background component if its spatial likelihood exceeds a predefined threshold T^s_{lik}:

$$log(p(\mathbf{x^s}|\theta^s_j)) > T^s_{lik} \tag{15}$$

A minimum probability threshold T_{map} is used to detect new objects and processes in the scene. The pixel is labeled "unassigned" in the support map if:

$$log(p(\mathbf{x}_t|\theta_{C\,map})) < T_{map} \tag{16}$$

We implement T_{map} by introducing a uniformly distributed component into the model. The density of this component is given by the extents of the feature space. For a frame size of 720×576 and YUV components in the range $[0, 1]$:

$$p(\mathbf{x_t}) = \frac{1}{720 \times 576} \tag{17}$$

This component has a fixed weight W_u, and pixels for which this is the most likely component are set as unassigned.

We also make an important performance optimization to the assignment process. If a pixel is currently assigned to an existing component, its color value remains relatively unchanged, and its probability given the same assignment is greater than T_{map} then we leave its assignment unchanged. This significantly reduces processing time, as we do not need to calculate the likelihood of each model component. A pixel value is defined as unchanged if each element of its YUV color value is within a threshold deviation from the value first used to assign it to the component. The result of this optimization is a significant increase in execution speed (approximately 5×), and an increase in the perceived stability of the algorithm. We have experimented with applying this optimization to all assigned pixels, and also to only those pixels assigned to background components, with similar results.

4.6. *Introducing Foreground Components*

All of the initial model components are labeled as background. Foreground components are introduced when regions of pixels appear which have a low probability under the mixture model. Such regions are taken to be generated by new foreground objects entering the scene, and appear in the support map as regions in which a high density of pixels have been labeled as unassigned by equation (16).

The support map is divided into a grid such that each cell has a resolution 16×16 pixels. The number of unassigned pixels is counted for each location. Locations exceeding a threshold density are considered to correspond to new foreground regions. A single foreground component is built from the statistics of all the unassigned pixels in these locations. The parameters of the component are estimated using equations of the form (10) to (13) to build the spatial and color distributions. This new component is then recursively split using the same procedure used to split background components. All new components introduced into the model in this way are initially labeled as foreground.

4.7. *Updating the Model*

After pixel assignment, the parameters of the existing background and foreground components are re-estimated. For foreground components, equations of the form (10) to (13) are used to calculate the spatial and color parameters from their assigned pixels.

For background components we adapt the parameters more slowly. For each component j we start by calculating a set of parameter values $\theta_{(j,sm)}$ from the new support map, in the same way as for foreground components. Given the previous parameters $\theta_{(j,t-1)}$, we calculate the new set $\theta_{(j,t)}$ using an adaptive learning rate:

$$\theta_{(j,t)} = \alpha_j \theta_{(j,sm)} + (1 - \alpha_j)\theta_{(j,t-1)} \tag{18}$$

Where α_j is a vector of learning rates, one for each model parameter, modified by a variable factor α_j^c such that:

$$\alpha_j = \alpha_j^c \left[\alpha_\mu^s, \alpha_\Sigma^s, \alpha_\mu^c, \alpha_\Sigma^c\right] \tag{19}$$

Where α_μ^s, α_Σ^s, α_μ^c, α_Σ^c are constants used to update the spatial mean and covariance, and color mean and covariance, respectively, and:

$$\alpha_j^c = \frac{\omega_{(j,sm)}}{\omega_{(j,t-1)}} \quad , \quad \alpha_j^c \in [0,1] \tag{20}$$

Where $\omega_{(j,sm)}$ and $\omega_{(j,t-1)}$ are the weights from $\theta_{(j,sm)}$ and $\theta_{(j,t-1)}$ respectively. Using α_j^c to factor the adaptation in this way ensures that if a background component is occluded it does not adapt too quickly to represent only the visible part. It also helps to prevent the background from over adapting to misclassified foreground pixels. It is necessary to renormalize the component weights at this point, to enforce the condition in equation (2).

Regardless of whether any new foreground components have been added this frame, we test all foreground components for possible merging. First, we restrict the spatial and color variances of each component to pre-defined maximum values. This helps prevent over adaptation to misclassified background pixels. We then merge similar components using the same pair-wise method as was used for the background model. We also examine foreground components for fragmentation, using a similar process to that used for detection and splitting of disconnected background components. This helps to maintain a one-to-one correlation between components and object processes. Finally, we conclude frame processing by culling any foreground components which have a zero or very low weight.

4.8. *Reclassifying Components*

All components introduced after model initialisation are classified as foreground. However, occasionally, a new foreground component will be introduced which does not correspond to foreground component, but to a change in the background process (for example, an illumination change). In such cases, the component classification is erroneous, and needs to be corrected.

We expect that foreground components will represent objects that are moving through the scene when they are first detected. Background components represent processes which may show some movement around an a mean position, but are relatively static.

We use this feature to detect inappropriately classified foreground components. In order to retain its classification, we impose the condition that a foreground component must exhibit a significant spatial translation immediately after instantiation. We implement this using two thresholds T_{fg}^s and T_{fg}^t such that if the the following condition is not satisfied, a foreground component is reclassified as background:

$$|\mu_{(j,t=0)} - \mu_{(j,t=T_{fg}^t)}| \geq T_{fg}^s \tag{21}$$

Where $t = 0$ corresponds to the image frame at which the component was created. The values used for T_{fg}^s and T_{fg}^t

are contextualized and reflect our expectations about foreground object behavior. Thus, unlike per-pixel methods, we can define the difference between foreground and background as an object-level attribute. We have experimented with various values of T^s_{fg} and T^t_{fg} in our experiments.

4.9. *Frame to Frame Object Correspondence*

Many systems (for example, [9]) implement frame-to-frame foreground object correspondence as a separate higher level process. In our system, correspondence is integrated with the model update process.

Assuming that foreground object movements from one frame to the next are relatively small, pixel values generated by a moving foreground object at time t will generally be assigned to the corresponding component $\theta_{(j,t-1)}$. Although spatial translation of the object decreases the component likelihood, the color likelihood will remain high. Thus object pixels are repeatedly re-assigned to the corresponding component(s), and the component parameters are re-estimated, such that the spatial mean of the component tracks the moving object.

Large object translations, or nearby similarly colored background components may cause pixels to be incorrectly assigned to a different component. In this case the existing corresponding foreground component will be extinguished, and a new component re-introduced automatically: the segmentation process is still effective, but correspondence is lost.

5. Experiments

We have performed a series of experiments to compare the segmentation quality of our model with Stauffer's per-pixel algorithm [23], and with Migdal's MRF based scheme [10]. We are particularly interested in the ability of our algorithm to extract foreground objects where the scene background exhibits unpredictable changes in spatial structure; however, we are also interested in general performance. We have therefore conducted evaluations against two separate data sets.

The first set comprises eleven video sequences filmed mainly in indoor environments in which the background is static: the only variations arise from slight changes in lighting conditions. Most sequences comprise one or two human targets performing routine actions such as walking and sitting down.

The second set comprises ten sequences which are more challenging. These have been filmed in outdoor sequences in which there is significant, and sometimes large, movement in the background. Some backgrounds comprise background foliage which are moved by wind. Others contain moving water. In most sequences the foreground target is human, though we have also included a sequence in which the target is a car, and in another, a moving bird. Some example frames from sequences in the second data set are shown in the left hand column of figure 2.

All sequences were filmed using a standard consumer DV camcorder producing a PAL format video stream (720 $\times$ 576 pixel frame size, at 25 Hz, interlaced). The captured sequences were re-sampled to a frequency of 10Hz (by omitting frames), and a simple de-interlacing algorithm was applied. The duration of the processed sequences ranged between 50 and 500 frames. For each of the algorithms we used parameter ranges which ensured that the background model was learned robustly well within the minimum sequence duration (see section 5.1).

5.1. *Quantifying Performance*

In order to quantify the performance of the three algorithms we constructed a set of "ground truth" frames for each of the sequences. From each we arbitrarily selected a sample of sixteen frames: to avoid bias, the frames were selected without prior inspection. We also avoided selecting frames from the beginning each sequence, so that the algorithms were able to properly initialise their background models. These frames were then copied, and the copies manually annotated by marking the foreground pixels as pure red (RGB 255,0,0) in an image editing program. These annotated frames are considered to represent a "correct" ground truth segmentation of the corresponding frames.

For each of the three algorithms we performed the following automated procedure. The algorithm was run over each sequence, generating a set of foreground pixels for each frame. For our algorithm, this corresponds to the set of pixels assigned to foreground components. For frames which have a ground truth, the algorithm output was compared to the manual segmentation. A pixel classified as foreground by both the algorithm and the annotation is denoted "true positive" (TP) foreground. If it classified as foreground by only the algorithm, it is considered "false positive" (FP). Finally, if it classified as foreground by only the annotation then it is considered "false negative" (FN). The total number of TP, FP, and FN pixels is summed for for each sequence, resulting totals for each algorithm against each sequence. We avoided choosing ground truth frames near the beginning of each sequence, so that the algorithms were able to initialize properly before we examined their output.

We use the TP, FP, and FN values for each sequence (and summed for all sequences in each data set) to construct two different comparison metrics. A range of metrics is presented by Rosin [17], including Jaccard coefficient J_c, where:

$$J_c = \frac{TP}{(TP + FP + FN)} \tag{22}$$

This metric was also used by Migdal [10], and so we use it to represent our results. For our second metric we use the total error, E_{tot}, used by Toyama [25], where:

$$E_{tot} = FP + FN \tag{23}$$

Parameter	Value
Max. Background Components	1000
Max. Foreground Components	300
Background split (spatial) T^s_{sp}	800
Background split (color) T^c_{sp}	50
Background merge (spatial) T^s_{mg}	2
Background merge (color) T^c_{mg}	1
Background update rate (Color) α^c_μ	0.05
Background update rate (Spatial) α^s_μ	0.0 (no update)
Uniform Component Weight W_u	0.1
Foreground reclassification time T^t_{fg}	2.0s
Foreground reclassification distance T^s_{fg}	48 pixels

Table 1
Most Effective Parameter Values for our Algorithm (Data Set 2)

Parameter	Value
Number of components per pixel	3
Learning rate	0.02
Match threshold	3.0

Table 2
Most Effective Parameter Values for Stauffer's Algorithm (Data Set 2)

Parameter	Value
Learning rate (component weight)	0.005
Learning rate (component parameters)	0.05
Start temperature	1.0
End temperature	0.2

Table 3
Most Effective Parameter Values for Migdal's Algorithm (Data Set 2)

We repeated our experiments with a range of parameters for each algorithm. In the case of Migdal's algorithm we experimented with both the proposed linear cooling schedule, and an exponential schedule more commonly used to estimate MAP-MRF field labelings. For our algorithm, the parameters which effect performance were found to be the number of components used for the model, the model update rate, and the parameters used to reclassify foreground components as background. A summary of the best parameter values found for the second data set is shown in table 1. We compared results using this parameter set against the best parameters found for both Stauffer's and Migdal's algorithms. The best parameters for these are shown in tables 2 and 3 respectively.

5.2. *Results*

Results for the first data set, with static backgrounds, are shown in table 4. These table shows Jaccard coefficients and total errors for each sequence, and totals for the whole set. Both metrics indicate that the MRF segmentation gives a better overall performance than our algorithm, with both giving much better results than Stauffer's model. Migdal's scheme has the highest Jaccard coefficient for ten of the eleven sequences, and the lowest error rate in seven cases. Our algorithm has the highest Jaccard coefficient for one sequence, and the lowest error rate in three sequences.

	Jaccard Coefficient			Total Errors ($\times 10^3$)		
Sequence	Stauffer	Migdal	Ours	Stauffer	Migdal	Ours
1	0.54	**0.83**	0.68	244	**76**	196
2	0.63	**0.84**	0.72	333	**109**	174
3	0.48	**0.84**	0.73	354	**79**	147
4	0.19	0.31	**0.32**	**374**	577	386
5	0.54	**0.62**	0.52	237	202	**164**
6	0.53	**0.78**	0.65	98	**38**	52
7	0.39	**0.48**	0.45	194	182	**150**
8	0.23	**0.63**	0.56	890	**221**	313
9	0.65	**0.70**	0.60	434	350	**264**
10	0.68	**0.84**	0.79	162	**59**	122
11	0.36	**0.52**	0.44	194	**165**	187
total	0.47	**0.66**	0.59	3515	**2060**	2154

Table 4
Jaccard Coefficients and Total errors ($\times 10^3$) for Scenes with Static Backgrounds (Data Set 1)

Although Migdal's algorithm gives a better performance on the first data set, results for the second data set, with dynamic backgrounds, are very different. These are shown in table 5. On this data set, our algorithm outperforms the MRF and per-pixel models by a considerable margin, with the highest Jaccard coefficient in nine out of ten cases, and a lower error rate in all cases. To visualize, Figure 2 shows example image frames and segmentations from six of the ten sequences. In each case, the original frame is shown in the left hand column, the per-pixel segmentation in the next column, the MRF segmentation in the next, and the result for our algorithm in the right hand column. In all cases, the background exhibits significant spatio-temporal variation during the sequence.

Comparing the results for the two data sets, we see that variations in the background, caused by moving foliage or water, drastically reduce the effectiveness of both the MRF and per-pixel models whilst our algorithm remains robust. The total Jaccard coefficient for the MRF scheme reduces from 0.66 for the first data set to only 0.15 for the second. For Stauffer's algorithm, it is reduced from 0.44 to 0.07. In both cases, the segmentations for the second data set are poor. For our algorithm, performance drops only a little, from 0.59 to 0.52. Based on these results we assert that our algorithm gives a better segmentation than either Stauffer's or Migdal's in scenes with significant levels of background movement.

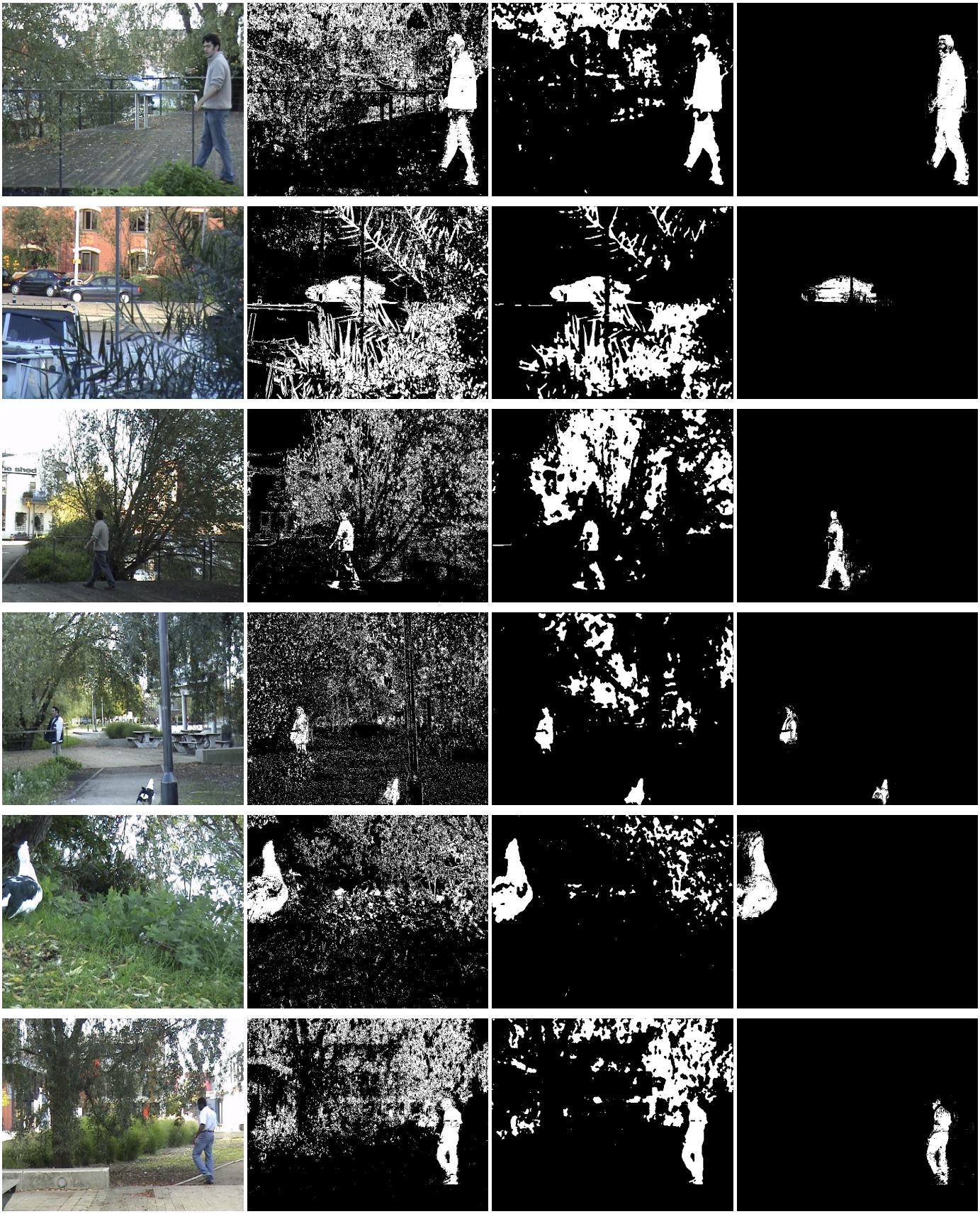

Fig. 2. Segmented foreground pixels. Left to right : Original image, Stauffer, Migdal, Our algorithm

	Jaccard Coefficient			Total Errors ($\times 10^3$)		
Sequence	Stauffer	Migdal	Ours	Stauffer	Migdal	Ours
1	0.10	0.15	**0.58**	1322	1100	**109**
2	0.04	0.04	**0.52**	1642	1529	**77**
3	0.14	**0.52**	0.34	546	131	**127**
4	0.01	0.23	**0.33**	964	567	**14**
5	0.07	0.21	**0.58**	886	250	**40**
6	0.05	0.08	**0.61**	1146	797	**46**
7	0.14	0.25	**0.58**	1237	671	**181**
8	0.09	0.28	**0.45**	1000	348	**118**
9	0.02	0.01	**0.46**	981	382	**187**
10	0.09	0.21	**0.53**	933	328	**95**
total	0.07	0.15	**0.52**	10657	5594	**827**

Table 5
Jaccard Coefficients and Total errors ($\times 10^3$) for Scenes with Dynamic Backgrounds (Data Set 2)

5.3. *Significance of the Results*

In this section we verify that the differences in performance apparent in the previous section are statistically significant.

We first consider the results of our algorithm against Stauffer's on the second data set. Our hypothesis is that our algorithm gives a better segmentation (referenced against the ground truth) than Stauffer's, on sequences with a high degree of background movement. Our null hypothesis, H_0^1, is that *there is at least a 0.5 probability that Stauffer's algorithm will return a higher Jaccard coefficient than ours on an image sequence*. This hypothesis asserts that, using the Jaccard coefficient as a performance metric, Stauffer's algorithm will out-perform ours on average. We consider the result on each sequence as a Bernoulli trial with $p = 0.5$ and use the binomial distribution to determine an appropriate p-value. In this case our algorithm returns a higher Jaccard coefficient for all ten sequences, resulting in a p-value of $0.5^{10} \approx 0.001$. We can therefore reject H_0^1 at the standard 5% significance level. Similarly, we reject the null hypothesis regarding total errors, H_0^2, that *there is at least a 0.5 probability that Stauffer's algorithm will return a lower total error than ours*, at the same significance level.

Applying the same procedure to the results against Migdal's algorithm, we construct two similar null hypotheses H_0^3 and H_0^4. Our algorithm returns a higher Jaccard coefficient than Migdal's in all but one case. For H_0^3 this equates to a p-value of $11 \times 0.5^{10} \approx 0.011$. Again, this is well within than standard 5% significance level, and we can reject H_0^3. Our algorithm returns a lower total error than Migdal's in all cases, and so, as with H_0^2, we can reject H_0^4 at the 5% significance level.

We proceed to consider the significance of our results against data set 1, shown in table 4. In this dataset, our algorithm gave a higher Jaccard coefficient that Stauffer's in nine out of 11 sequences, and a lower error rate in ten cases.

Algorithm	Approx. frame processing time (s)
Stauffer	0.1
Migdal	5.0
Our algorithm	2.0 - 4.0

Table 6
Approximate image frame processing times, in seconds

Following the same procedure as for data set 2, we propose the same null hypotheses for data set 1: H_0^5 and H_0^6 which correspond to H_0^1 and H_0^2 respectively. Corresponding p-values are $67 \times 0.5^{11} \approx 0.033$ and $12 \times 0.5^{11} \approx 0.006$, and we can reject both at the 5% significance level.

We use a different treatment for our results against Migdal's algorithm on data set 1, since Migdal's algorithm returns a better Jaccard coefficient in ten cases, and lower error rate in 7 cases. We therefore propose alternative null hypotheses to validate the superiority of Migdal's algorithm on sequences with static backgrounds, denoted H_0^7 and H_0^8 respectively: *there is at least a 0.5 probability that our algorithm will return a higher Jaccard coefficient than Migdal's*; *there is at least a 0.5 probability that our algorithm will return a lower total error than Migdal's*. Corresponding p-values are $12 \times 0.5^{11} \approx 0.001$ and $552 \times 0.5^{11} \approx 0.27$. We therefore reject H_0^7 at the 5% significance level, and conclude that Migdal's algorithm returns a better Jaccard coefficient for scenes with static backgrounds. However, we are unable to reject H_0^8, so our comparison of error rates is inconclusive in this case.

We have thus shown that our results are statistically significant except in one case: the comparison of total error rates between our algorithm and Migdal's on data set 1, supporting our assertions in the previous section.

5.4. *Frame Processing Time*

All three algorithms were coded in unoptimized C++ and our experiments were run on a 2.8 GHz Pentium 4 based PC. As previously mentioned, the processed frame size is 720×576. The execution times per frame are shown in table 6. From this table it can be seen that Stauffer's algorithm is considerably faster than ours or Migdal's, trading segmentation accuracy for speed. Migdal's algorithm is the slowest, whereas execution time for our algorithm varies depending on the number of components and the relative change in scene structure between frames.

The results reported in table 6 suggest that, whilst our algorithm is significantly faster than Migdal's MRF scheme, neither scheme is yet capable of real-time performance. In comparison with Stauffer's algorithm it is clear that there is a significant computational cost to enforcing spatial coherency, using either method. However, we believe that there is considerable scope for improving the frame execution times that we have reported: firstly through optimization of our C++ implementation, and secondly by the use of specialized hardware.

6. Conclusions

The processes observed in real-world scenes are complex and chaotic, and often make accurate background subtraction a challenging task. We have considered the problem of robust segmentation in scenes with dynamic backgrounds: where objects in the background are subject to spatial variations. Per-pixel background models are unable to effectively represent such variations, leading to frequent mis-classifications. Random field models are able to impose global spatial and temporal dependencies on field labelings, but do not explicitly model the spatial structure of the scene.

We have proposed a scheme in which homogeneous regions of the scene are modeled by an adaptive mixture of Gaussians in color and space. Components of the model represent clusters of pixels generated by discreet processes or objects in both the background and foreground. We use this model to probabilistically classify new pixel observations, and remove misclassifications caused by spatio-temporal background variations.

We conducted a series of experiments to investigate the effectiveness of our model, and compare its performance to per-pixel and MRF-based algorithms. We tested the algorithms on two data sets, the first comprising sequences with static backgrounds, and the second comprising scenes in which the background exhibited significant structural variations.

Our results show that whilst all three models are able to produce effective segmentations when the background is static, the output of the per-pixel and MRF based models is severely degraded by background variations. Our model is robust, however, and the segmentation quality was only marginally reduced. Having demonstrated the statistical significance of our results, there is strong evidence that explicitly modeling spatial features of the background results in a more robust segmentation.

There is a trade off between accuracy and processing time. A more complex model requires more processing, and both our system and the MRF-based system we tested were an order of magnitude slower than the per-pixel model. This is an issue for real-time monitoring systems; however, we believe that there is scope for speed optimizations, which would make our system a viable proposition for real-time implementation.

7. Acknowledgments

The work presented in this article was supported by an EPSRC CASE studentship (reference GP-P04329-01), in conjunction with Nectar Electronics Ltd. UK. The authors would like to thank Ray Broadbridge of Nectar Electronics for contributions in supporting this work.

References

[1] J. Cheng, J. Yang, Y. Zhou, and Y. Cui. Flexible background mixture models for foreground segmentation. *Image and Vision Computing*, 24(5):473–482, 2006.

[2] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis. Background and foreground modeling using nonparametric kernel density estimation for visual surveillance. *Proceedings of the IEEE*, 90(7):1151–1163, July 2002.

[3] H. Greenspan, J. Goldberger, and A. Mayer. Probabilistic spacetime video modeling via piecewise GMM. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 26(3):384–396, 2004.

[4] I. Haritaoglu, D. Harwood, and L. Davis. W4: Who? when? where? what? A real time system for detecting and tracking people. In *Proc. of International Conference on Automatic Face and Gesture Recognition*, pages 222–227, Nara, Japan, 1998.

[5] M. Harville, G. Gordon, and J. Woodfill. Foreground segmentation using adaptive mixture models in color and depth. In *Proc. of IEEE Workshop on Detection and Recognition of Events in Video*, pages 3–11, Vancouver, Canada, July 2001.

[6] B. Heisele. Motion-based object detection and tracking in color image sequences. In *Proc. of Asian Conference on Computer Vision*, pages 1028–1033, Taipei, Taiwan, 2000.

[7] C. Hua, H. Wu, Q. Chen, and T. Wada. A pixel-wise object tracking algorithm with target and background sample. In *Proc. of International Conference on Pattern Recognition*, volume 1, pages 739–742, Hong Kong, China, 2006.

[8] P. KaewTraKulPong and R. Bowden. An improved adaptive background mixture model for real-time tracking with shadow detection. In *In Proc. European Workshop on Advanced Video Based Surveillance Systems*, Kingston, UK, September 2001.

[9] S. McKenna, S. Jabri, Z. Duric, A. Rosenfeld, and H. Wechsler. Tracking groups of people. *Computer Vision and Image Understanding*, 80(1):42–56, 2000.

[10] J. Migdal and W. Grimson. Background subtraction using Markov thresholds. In *Proc. of IEEE Workshop on Applications of Computer Vision / IEEE Workshop on Motion and Video Computing*, volume 2, pages 58–65, Breckenridge, CO, USA, January 2005.

[11] A. Monnet, A. Mittal, N. Paragios, and V. Ramesh. Background modeling and subtraction of dynamic scenes. In *Proc. of IEEE International Conference on Computer Vision*, volume 2, pages 1305–1312, Nice, France, 2003.

[12] N. Oliver, B. Rosario, and A. Pentland. A Bayesian computer vision system for modeling human interactions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(8):831–843, 2000.

[13] N. Paragios and V. Ramesh. A MRF-based approach for real-time subway monitoring. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1034–1040, Hawaii, USA, 2001.

[14] A. Pece. Tracking by cluster analysis of image differences. In *Proc. of International Symposium on Intelligent Robotic Systems*, pages 295–303, Reading, UK, 2000.

[15] Y. Raja, S. J. Mckenna, and S. Gong. Color model selection and adaptation in dynamic scenes. In *Proc. of European Conference on Computer Vision*, volume 1, pages 460–474, Freiberg, Germany, 1998.

[16] Y. Ren, C. Chua, and Y. Ho. Motion detection with nonstationary background. *Machine Vision and Applications*, 13(5):332–343, 2003.

[17] P. Rosin and E. Ioannidis. Evaluation of global image thresholding for change detection. *Pattern Recognition Letters*, 24(14):2345–2356, 2003.

[18] K. Schindler and H. Wang. Smooth foreground-background segmentation for video processing. In *Proc. of Asian Conference*

on Computer Vision, volume 2, pages 581–590, Hyderabad, India, 2006.

[19] M. Seki, T. Wada, H. Fujiwara, and K. Sumi. Background subtraction based on cooccurrence of image variations. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 65–72, Madison, WI, USA, 2003.

[20] Y. Sheikh and M. Shah. Bayesian modeling of dynamic scenes for object detection. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 27(11):1778–1792, 2005.

[21] A. Shimada, D. Arita, and R Taniguchi. Dynamic control of adaptive mixture-of-Gaussians background model. In *Proc. of IEEE International Conference on Video and Signal Based Surveillance*, Sydney, Australia, 2006.

[22] P. Spagnolo, T. D'Orazio, M. Leo, and A. Distante. Moving object segmentation by background subtraction and temporal analysis. *Image and Vision Computing*, 24(5):411–423, 2006.

[23] C. Stauffer and W.Grimson. Adaptive background mixture models for real-time tracking. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 2, pages 246–252, Fort Collins, CO, USA, 1999.

[24] Y. Tian, M. Lu, and A. Hampapur. Robust and efficient foreground analysis for real-time video surveillance. In *Proc. of IEEE Conference on Computer Vision and Pattern Recognition*, volume 1, pages 1182–1187, San Diego, CA, USA, 2005.

[25] K. Toyama, J. Krumm, B. Brumitt, and B. Meyers. Wallflower: Principles and practice of background maintenance. In *Proc. of International Conference on Computer Vision*, volume 1, pages 255–261, Corfu, Greece, 1999.

[26] Y. Wang, K. Loe, T. Tan, and J. Wu. A dynamic hidden Markov random field model for foreground and shadow segmentation. In *Proc. of IEEE Workshop on Motion and Video Computing*, pages 474–480, Breckenridge, CO, USA, 2005.

[27] Y. Wang, K. Loe, and J. Wu. A dynamic conditional random field model for foreground and shadow segmentation. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 28(2):279–289, 2006.

[28] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland. Pfinder: Real-time tracking of the human body. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19(7):780–785, 1997.