Background Subtraction with Dirichlet Process Mixture Models

Tom S. F. Haines and Tao Xiang

Abstract—Video analysis often begins with background subtraction. This problem is often approached in two steps - a background model followed by a regularisation scheme. A model of the background allows it to be distinguished on a per-pixel basis from the foreground, whilst the regularisation combines information from adjacent pixels. We present a new method based on Dirichlet process Gaussian mixture models, which are used to estimate per-pixel background distributions. It is followed by probabilistic regularisation. Using a non-parametric Bayesian method allows per-pixel mode counts to be automatically inferred, avoiding over-/under-fitting. We also develop novel model learning algorithms for continuous update of the model in a principled fashion as the scene changes. These key advantages enable us to outperform the state-of-the-art alternatives on four benchmarks.

Index Terms—Background subtraction, Dirichlet processes, non-parametric Bayesian methods, confidence capping, video analysis

1 INTRODUCTION

Background subtraction can be defined as a binary segmentation of a video stream, into the foreground, which is unique to a particular moment in time, and the background, which is always present. It is typically used as an interest detector for higher level problems, such as automated surveillance, action recognition, intelligent environments and motion analysis. The etymology of background subtraction derives from the original approach, where a single static image of just the background is subtracted from the current frame, to generate a difference image. If the absolute difference exceeds a threshold the pixel in question is declared to belong to the foreground. This approach performs poorly because it assumes a static background with well behaved objects - in practise this is almost never the case. Many situations break these assumptions [1], [2]:

- Dynamic background, where objects such as trees blow in the wind, escalators move and traffic lights change colour. These objects, whilst moving, still belong to the background as they are of no interest to further analysis.
- Noise, as caused by the image capturing process. It can vary over the image due to photon noise and varying brightness. In some cases, such as low light/thermal, it can dominate.
- Camouflage, where a foreground object looks very much like the background, e.g. a sniper in a ghillie suit.
- Camera shake often exists, a symptom of mount points that are subject to wind or vibrations. This can be considered to be a type of highly correlated global noise.

Moved object (class transitions), where the foreground becomes part of the background, or vice versa; e.g. a car could be parked in the scene, and after sufficient time considered part of the background, only to later become foreground again when driven off.

Bootstrapping. As it is often not possible to get a frame with no foreground an algorithm should be capable of being initialised with foreground objects in the scene. It has to then learn the correct background model over time.

Shadows are cast by the foreground objects, but later processing is typically not interested in them.

Illumination changes, both gradual, e.g. from the sun moving during the day, and rapid, such as from a light switch being toggled.

Addressing these challenges, particularly the first three, leads to a number of considerations in designing a background model. Specifically, on account of the dynamic nature of backgrounds the model has to classify a range of pixels as belonging to the background. This can be represented with a multi-modal distribution, e.g. a tree waving against the sky generates pixels that are sky one moment and tree another, but both sets need to be modelled as background. How many modes exist will need to be learnt, and can potentially change with time, e.g. a windy day versus a calm day.

Noise is generated by both the image capture process and typical small scale variations in the background colour. It is primarily modelled using the variance of a probability distribution, which needs to be set high enough to capture the variation. Camouflage on the other hand is when a foreground object looks like it belongs to the background, either deliberately or accidentally. The effect of camouflage is interleaved with that of noise: noise can increase the range of values considered to belong to the background, allowing a camouflaged object to remain undetected. Consequently the variance of the background representation is a trade-off - too low and...
noise will trigger detection; too high and camouflaged objects will be missed. Variance therefore needs to be learnt from the data. Noting that as noise can vary with both location and time (e.g. photon noise only matters in the dark areas of an image, at night.), this needs to be continuously calculated from the video feed, on a per-region basis. A desirable background model thus should learn the number of modes required to represent the background, including the variance of each mode, such that it handles noise without compromising its ability to detect camouflaged entities.

To this end we propose to use a Dirichlet process Gaussian mixture model (DP-GMM) \(^1\) to provide a per-pixel density estimate. This is a non-parametric Bayesian method \(^2\) that automatically estimates the number of mixture components required to model the pixels background colour distribution, e.g. a tree waving backwards and forward in front of the sky will generate single mode pixels at the trunk and in the sky, but two mode pixels in the area where the branches wave, such that the pixels transition between leaf and sky regularly. If it needs more modes to represent multiple leaf colours this will occur automatically, and, of great importance for long term surveillance, the model will update with time as new evidence becomes available, e.g. on a calm day when the tree is not moving it will revert to single mode pixels throughout. As it is fully Bayesian it avoids over-/under-fitting its model.

However, there are two issues that prevent a standard DP-GMM from being used as a per-pixel background model: 1) the existing model update techniques cannot cope with the scene changes common in real-world applications; 2) using the model for continuous video streams is unscalable both in terms of memory usage and computational cost. To solve the first issue and make the DP-GMM suitable for background modelling, a set of novel model update techniques are developed in this work. More specifically, each mode in the model represents a range of colours, using a Gaussian distribution \(^3\). The extent of this range is learnt from the data stream, and is again updated as the scene changes. During the day when pixels are almost constant it will select a tight distribution, whilst at night it will expand as photon noise pushes the range of expected values wider. A visual scene typically evolves as time goes by, e.g. the day/night cycle, a car parking, road works remodelling a traffic junctions layout, with different speeds of change. Consequently the model needs to forget what it has previously learnt and model the new background effectively and efficiently. For this we introduce a novel model update concept that lets old information degrade in a principled way, which we refer to as confidence capping. It limits how confident it can be about any given mode, which puts a constraint on how certain it can be about the shape of the background distribution. As a result, when a mode is no longer used it will slowly fade away, whilst the new mode(s) start to dominate the model. If during this period the scene reverts to the old state it can start using its old modes again. This allows a stationary object to remain part of the foreground for a long time, as it takes a lot of information for the new component to obtain the confidence of pre-existing components, but when an object moves on and the background changes to a component it has seen before, even if a while ago, it can use that component immediately. Updating the components for gradual background changes (e.g. gradual illumination changes) continues to happen, making sure the model is never left behind by gradual changes. This also helps with bootstrap performance, as it can quickly learn the model, even with foreground objects.

To overcome the second issue on model scalability, two measures are taken. First, we fit the model by Gibbs sampling each sample only once. This would normally compromise model convergence, but because of confidence capping and the availability of a never ending stream of data the model will converge regardless. In addition to its computational saving it avoids the need to store and process hundreds of previous video frames, which can consume vast amount of memory \(^4\), \(^5\). This memory consumption problem is particularly noticeable for approaches based on kernel density estimation \(^6\), \(^7\), \(^8\), \(^9\), \(^10\). Second, to obtain real time performance, we introduce a GPU based implementation.

To summarise the contributions are: 1) a new background subtraction technique is proposed \(^11\) built on a non-parametric Bayesian density estimate, which avoids over-/under-fitting; 2) A novel model update is formulated that is both principled and practical - the approach is real time with a constant memory requirement; 3) An exhaustive evaluation using four data sets \(^11\), \(^2\), \(^12\), \(^13\) demonstrates top performance in comparison with the state-of-the-art alternatives. The code, both a C version and a GPU version, has been made available.

1.1 Related Work

The background subtraction field is humongous, and has many review papers \(^2\), \(^14\), \(^15\), \(^16\). Stauffer & Grimson \(^17\) is one of the best known approaches - it uses a Gaussian mixture model (GMM) for a per-pixel density estimate (DE) followed by connected components for regularisation. This model improves on using a background plate because it can handle a dynamic background and noise, by using multimodal probability distributions. As it is continuously updated it can bootstrap. Its mixture model includes both foreground and background components - it classifies values based on their mixture component, which is assigned to the foreground or the background based on the assumption that the majority of the larger components belong to the background, with the remainder foreground. This

---

1. This is actually integrated out, such that a student-t distribution is used in practise.

2. Accessible at [http://www.thaines.com](http://www.thaines.com)
assumption fails if objects hang around for very long, as they quickly dominate the distribution. It has a fixed component count, which does not match real variability, where the number of modes differs between pixels. The model is updated linearly using a fixed learning rate parameter - consequentially the moved object problem is an unhelpful trade between ghosting and forgetting stationary objects too quickly. Connected components converts the intermediate foreground mask into regions via pixel adjacency, and culls all regions below a certain size, to remove spurious detections. This approach to noise handling combined with its somewhat primitive density estimation method undermines camouflage handling, as it often thinks it is noise, and also prevents it from tracking small objects. No capacity exists for it to handle illumination changes or shadows. The above can be divided into four parts - the model, updating the model, how pixels are classified, and regularisation; alternative approaches for each will now be considered in turn. 

**The model:** Alternative DE methods exist, including different GMM implementations and kernel density estimate (KDE) methods, either using Gaussian kernels or step kernels. Histograms have also been used, and alternatives to DE include models that predict the next value, use neural networks, or hidden Markov models. Alternatives to the GMM background model can improve handling of dynamic background, noise and camouflage by better reflecting the underlying variability of the background. This can be particularly important with relation to over-/under-fitting, as the model needs to accurately generalise the data stream. All models suffer from some degree of over-/under-fitting. KDE and histogram methods are particularly vulnerable, as they implicitly assume a constant density by using fixed size kernels/bins. GMM methods should do better, but the heuristics required for online learning, particularly regarding the creation of new components, can result in local minima in the optimisation, which is just as problematic. He et al. recently used DP-GMMs for background subtraction, in common with the presented approach. They failed to leverage the potential advantages however (discussed below), and used computationally unscalable methods. It is block-based, which is to say they ran the model at very low resolution to save computation, and it has unbounded memory consumption. Consequentially it is not applicable to real world scenarios, and, despite their efforts, poor results were obtained. Global models should in principal be better, with techniques based on principal component analysis (PCA) having shown the most promise. They attempt to model the background as a low dimensional subspace of the vectorised input, with the foreground identified as outliers. In practise such approaches have struggled, due to high computational requirements and limited capability to deal with many common problems, e.g. camouflage. They tend to excel at camera shake however, and recent variants have resolved many of the issues.

**Model update:** Most methods use a constant learning rate to update the model, but some use adaptive heuristics, whilst others are history based, and build a model from the last n frames directly. Adapting the learning rate affects the moved object issue - if it is too fast then stationary objects become part of the background too quickly, if it is too slow it takes too long to recover from changes to the background. Adaptation aims to adjust the rate depending on what is happening. There is invariably a trade-off between quickly learning when the background model has changed and accepting foreground objects into the background when they stop moving for a while (cars at traffic lights.) - trying to minimise this trade-off is a key goal and no perfect solution exists. In addition continuously learning the model is required to handle the bootstrapping issue. We present confidence capping (see above), which works because non-parametric Bayesian models, such as DP-GMMs, have a rigorous concept of a new mixture component forming. Parametric models have to use heuristics to simulate this, whilst a confidence cap is not possible for KDE based approaches as they lack a measure of confidence.

**Pixel classification:** The use of a single density estimate that includes both foreground (fg) and background (bg), as done by Stauffer & Grimson is somewhat unusual - most methods stick to separate models and apply Bayes rule, with the foreground model set to be the uniform distribution as it is unknown. We follow this convention, which results in a probability of being bg or fg, rather than a hard classification, which is passed through to the regularisation step. Instead of using Bayes rule some works use a threshold, which can by dynamically learnt. Attempts at learning a foreground model also exist, and some models generate a binary classification directly.

**Regularisation:** Some approaches have no regularisation step, others have information sharing between adjacent pixels but no explicit regularisation. Techniques such as eroding then dilating are common, and more advanced techniques have, for instance, tried to match pixels against neighbouring pixels, to compensate for background motion. When dealing with a probabilistic fg/bg assignment probabilistic methods should be used, such as the use of Markov random fields (MRF) by Migdal & Grimson, Sheikh & Shah and Schick et al. We also use a MRF - the pixels all have a random variable which can take on one of two labels, fg or bg. The data term is provided by the model whilst pairwise potentials encourage adjacent pixels to share the same label. Differences exist - previous works use Gibbs sampling and graph cuts, whilst we choose loopy belief propagation, as run time can be capped, and also accelerated with a GPU implementation. We also use an edge preserving cost between pixels, rather than a constant cost, which proves to be beneficial with high levels of noise. Cohen has also used a
Markov random field, to generate a background image by selecting pixels from a frame sequence, rather than for regularisation. Parks & Fels\cite{32} provide a review of regularisation for background subtraction.

**Input:** The vast majority of approaches, including this one\cite{11}, use the colours of the individual pixels directly. Other approaches use alternative information sources however, including optical flow\cite{33}, depth\cite{34}, tracking\cite{35} and object detection\cite{35}. Additionally, only foreground/background classification is provided by the presented approach - other approaches may mark areas as being in shadow\cite{6},\cite{35} or use other labelling strategies\cite{33}.

### 2 Methodology

The presented method naturally splits into two - a per pixel background model and a regularisation step. The GPU implementation and further details are also given. For an overview of the system Figure 1 gives a block diagram, whilst Algorithm 1 gives pseudo code of the core structure.

#### 2.1 Per-pixel Background Model

Each pixel has its own multi-modal density estimate, used to model \( P(x|bg) \) where \( x \) is the vector of pixel colour channels. The Dirichlet process Gaussian mixture model (DP-GMM)\cite{3} is used. It can be viewed as the Dirichlet distribution extended to an infinite number of components, which allows it to learn the true number of mixtures required to represent the data. For each pixel a stream of values arrives, one with each frame - the model has to be continuously updated using incremental learning.

**Algorithm 1** Pseudo code for core algorithm

(Actual code available from [http://www.thaines.com](http://www.thaines.com))

```plaintext
for frame in video:
    frame = colour_convert(frame) # Subsection 2.4 & supplementary material.
model.update_global_prior(frame) # Subsection 2.4
model.correct_lighting(frame) # Subsection 2.4
for x,y in frame.coordinates():
    pixel = frame[x,y]
    pixel_sd = camera_shake(frame, x, y) # Eqn. 17
    for t in [components] + [new]:
        p_com[x,y,t] = model.p_draw_from_component(pixel, t) # Equation 9
        p_bg[x,y] = sum(p_com[x,y,:]) / 3.0 # Equation 12 with P(bg) = 0.5
    t = draw(p_com[x,y,:]) # Weighted draw
model.update(x,y,t) # Equations 4 - 6 & 16
model.cap_confidence(x,y) # End of subsection 2.1
p_bg = belief_propagation(p_bg) # Regularisation, sub section 2.3
mask = threshold(p_bg)
```

![Fig. 1. Block diagram of the approach. The solid arrows show the path of a frame through the system, whilst the dotted arrows show the update order of the model, which loops around for the next frame.](image)

We start by introducing the Dirichlet process, firstly using the stick breaking construction then secondly using the Chinese restaurant process. The stick breaking construction provides a clean explanation of the concept, whilst the Chinese restaurant process integrates out irrelevant variables and provides the formulation we actually solve. Figure 2a represents the DP-GMM graphically using the stick breaking construction. It can be split into 3 columns - on the left the priors, in the middle the entities representing the Dirichlet process (DP) and on the right the data for which a density estimate is being constructed. This last column contains the feature vectors (pixel colours) to which the model is being fitted, \( x_i \), which come from all previous frames, \( i \in I \). It is a generative model - each sample comes from a specific mixture component, indexed by \( Z_i \in \mathcal{K} \), which consists of its probability of being selected, \( V_k \) and the Gaussian distribution from which the value was drawn, \( \eta_k \). The conjugate prior, consisting of \( \mu \), a Gaussian over its mean, and \( \Lambda \), a Wishart distribution over its precision (inverse covariance matrix), is applied to all \( \eta_k \). So far this is just a mixture model; the interesting part is that \( \mathcal{K} \), the set of mixture components, is infinite. Conceptually the stick breaking construction is very simple - we have a stick of length 1, representing the entire probability mass, which we keep breaking into two parts. Each time it is broken one of the parts becomes the probability mass for a mixture component - a value of \( V_k \), whilst the other is kept for the next break. This continues forever. \( \alpha \) is the concentration parameter, which controls how the stick is broken - a low value puts most of the probability mass in a few mixture components, whilst a high value spreads it out over many. Stick lengths are represented by the
**CRP** is used for the implementation of the conjugate prior over the Gaussian.

Each stick is associated with a draw, \( k \), where \( V_k \) represents the remaining stick length, calculated from the remaining stick minus one drawn. In effect, the infinite set of sticks is broken down into a computationally tractable form. This is impractical however as historic data would need to be kept.

**Fig. 2.** Two representations of the DP-GMM graphical model, each corresponding to a specific implementation technique.

---

**Random variable** \( V_k \), which is defined as

\[
V_k = v_k \prod_{i \in (0,k)} (1 - v_i), \quad v_k \sim \text{Beta}(1, \alpha)
\]

where \( \text{Beta}(\cdot, \cdot) \) is the beta distribution and \( k \in K \) is the infinite set of sticks. In effect, the \( v_k \) values are how much of the remaining stick to use for probability mass \( k \), whilst \( V_k \) is the remaining stick length, calculated from the previous breaks, multiplied by the amount of the remainder to use for stick \( k \). Orthogonal to the stick length, each stick is associated with a draw, \( \eta_k \), from the DP’s base measure, which is the already mentioned conjugate prior over the Gaussian.

Whilst the stick breaking construction offers a clean explanation of the model, the Chinese restaurant process (CRP) is used for the implementation. This is the model with the middle column of Figure 2b integrated out, to give Figure 2c. It is named by an analogy. Specifically, each sample is represented by a customer who either sits at a table or turns up and sits at a table in a Chinese restaurant. Tables represent the mixture components, and a customer chooses to sit at a table where customers are already sitting, with probability proportional to the number of customers at that table, or to sit at a new table, with probability proportional to \( \alpha \). At each table (component) only one dish is consumed, which is chosen from the menu (base measure) by the first customer to sit at that table. Integration out the draw from the DP leads to better convergence, but more importantly replaces the infinite set of sticks with a computationally tractable finite set of tables. Whilst the theory behind Dirichlet processes is deep, the actual implementation of a CRP is fairly trivial, as it can be treated as the addition of a dummy table, representing the creation of a new table, to an otherwise standard mixture model. Note that when the dummy table is used a new one is made available.

The DP-GMM density estimate for each pixel is updated with each new frame. Updating proceeds by first calculating the probability of the current pixel value, \( x \), given the current background model, then updating the model with \( x \), weighted by the calculated probability - these steps are now detailed.

**Mixture components:** The per-pixel model is a set of weighted mixture components, such that the weights sum to 1, where each component is a Gaussian distribution. They are integrated out however, using the Chinese restaurant process for the mixture weights and the conjugate prior for the Gaussians (student-t distribution). Whilst the literature already details this second part it is included for completeness. \( x_i \in [0,1], i \in \{0,1,2\} \) represents the pixel’s colour, and independence is assumed between the components for reasons of speed (Equivalent to a covariance matrix where off diagonal entries are zero). This simplifies the Wishart prior to a gamma prior for each channel \( i \), such that

\[
\sigma_i^{-2} \sim \Gamma \left( \frac{n_i,0}{2}, \frac{\sigma_i^2}{2} \right), \quad \mu_i | \sigma_i^2 \sim \mathcal{N} \left( \mu_{i,0}, \frac{\sigma_i^2}{k_{i,0}} \right),
\]

where \( \mathcal{N}(\mu, \sigma^2) \) represents the normal distribution and \( \Gamma(\alpha, \beta) \) the gamma distribution. The parameters \( n_{i,0} \) and \( \sigma_{i,0} \), \( i \in \{0,1,2\} \), are the \( \Lambda \) prior from the graphical model, whilst \( \mu_{i,0} \) and \( k_{i,0} \) are the \( \mu \) prior.

Evidence, \( x \), is provided incrementally, one sample at a time, which will be weighted by \( w \), the probability that it belongs to the background model. The model is then updated from having \( m \) samples to \( m+1 \) samples using

\[
n_{i,m+1} = n_{i,m} + w,
\]

\[
k_{i,m+1} = k_{i,m} + w,
\]

\[
\mu_{i,m+1} = \frac{k_{i,m} \mu_{i,m} + w x_i}{k_{i,m} + w},
\]

\[
\sigma_{i,m+1}^2 = \sigma_{i,m}^2 + \frac{k_{i,m} w}{k_{i,m} + w} (x_i - \mu_{i,m})^2.
\]

Note that \( n_{i,m} \) and \( k_{i,m} \) have the same update, so one value can be stored to cover both, for all \( i \). Given the above parameters, updated with the available evidence, a Gaussian may be drawn to sample the probability of a colour being drawn from this mixture component.
Instead of drawing it the Gaussian is integrated out, to give

\[ x_i \sim T \left( n_{i,m}, \mu_{i,m}, \frac{k_{i,m} + 1}{k_{i,m} n_{i,m}} \sigma_{i,m}^2 \right), \quad (8) \]

where \( T(v, \mu, \sigma^2) \) denotes the three parameter student-t distribution. Consequently each mixture component is actually a student-t distribution, to avoid the problem of sampling the Gaussian distribution that is actually being represented.

**Background probability:** The Chinese restaurant process is used to calculate the probability of a pixel, \( x \in [0, 1]^3 \) belonging to the background (bg) model. It integrates out the stick weights, unlike the stick breaking construction, which is essential to obtaining good performance. The probability of \( x \) given component (table) \( t \in T \) is

\[ P(x|t, bg) = \sum_{i \in T} P(x|n_t, k_t, \mu_t, \sigma_t^2), \quad (9) \]

\[ P(x|n_t, k_t, \mu_t, \sigma_t^2) = \prod_{i \in \{0,1,2\}} T \left( x_i|n_{t,i}, \mu_{t,i}, \frac{k_{t,i} + 1}{k_{t,i} n_{t,i}} \sigma_{t,i}^2 \right), \quad (10) \]

where \( s_t \) is the number of samples assigned to component \( t \), and \( n_t, k_t, \mu_t, \sigma_t \) are the parameters of the prior updated with the samples currently assigned to the component. Note that \( s_t \) is the offset from the prior values for \( n_t \) and \( k_t \), so these latter two do not need to be stored. By including a dummy component, \( t = new \in T \), which represents creating a new component (sitting at a new table) with \( s_{new} = \alpha \), this is the Chinese restaurant process. The student-t parameters for this dummy component are the prior without update. Finally, the mixture components can be summed out

\[ P(x|bg) = \sum_{t \in T} P(x|t, bg). \quad (11) \]

The goal is to calculate \( P(bg|x) \), not \( P(x|bg) \), hence Bayes rule is applied,

\[ P(bg|x) = \frac{P(x|bg)P(bg)}{P(x|fg) + P(x|bg)}. \quad (12) \]

Note that pixels can only belong to either the background or the foreground (fg), hence the denominator. \( P(x|bg) \) is given above, leaving \( P(bg) \) and \( P(x|fg) \). \( P(bg) \) is an implicit threshold on what is considered background and what is considered foreground, and is thus considered to be a parameter. \( P(x|fg) \) is unknown and hard to estimate, so the uniform distribution is used, which is a value of 1, as the volume of the colour space is 1 (see supplementary material).

**Model update:** To update the model at each pixel the current value is assigned to a mixture component, which is then updated - \( s_t \) is increased and the posterior for the Gaussian updated with the new evidence (Equations 4 - 7). Assignment is done probabilistically, by randomly drawing a component, weighted by \( P(x|t, bg) \) (Equation 9). The possibility of assignment to a new mixture component is included - in the context of background subtraction this is the possibility that the pixel represents something we have not seen before. This is equivalent to Gibbs sampling the density estimate, except we only sample each value once, on arrival. In consequence samples do not need to be stored. Updates are weighted by their probability of belonging to the background (Equation 12), i.e. \( w \) is set to \( P(bg|x) \) in Equations 4 - 7 and \( s_t \) is incremented by \( P(bg|x) \). Sampling each value just once is not an issue, as the continuous stream of data combined with confidence capping means the model soon converges.

**Confidence capping:** A learning rate is not used; instead, unique to a DP-GMM, we introduce the notion of capping the confidence of the model. This can be interpreted as an adaptive update \([9], [26]\), but it is both principled and very effective. In effect we are building an online density estimate with the ability to selectively forget, allowing newer data to take over when the background changes. If this was not done the model would get more confident as time goes on - if it had seen the background in its current state for 3 days then it would require 3 days of data showing the background in a different state before the new model became dominant. Confidence capping effectively limits the confidence so that a new model can always replace the old one in a short amount of time; it also avoids numerical overflow and allows single-sample Gibbs sampling to converge. The cap is applied to the largest individual component, rather than the distribution as a whole - this avoids mixtures with more components being required to use wider distributions to compensate.

Mathematically it works by capping how high \( s_t \) can go. When the cap is exceeded by any \( s_t \) for a pixel then a multiplier is applied to all \( s_t \), scaling the highest \( s_t \) down to equal the cap. Note that \( s_t \) is tied to \( n_t \) and \( k_t \), so they are also adjusted. As \( \sigma_t^2 \) is dependent on \( k_t \) (it includes \( k_t \) as a multiplier) an update is avoided by storing \( \sigma_t^2/k_t \) instead. The effectiveness is such that it can learn the initial model with less than 30 frames of data yet objects can remain still for many minutes before being merged into the background. This does not impede the ability of the model to update as the background changes. Finally, in the limit as an infinite number of frames arrives the Dirichlet process will have an infinite number of components. This will exhaust a computer’s memory; therefore the component count is also capped. When a new component is created the existing component with the lowest \( s_t \) is replaced.

### 2.2 Probabilistic Regularisation

The per-pixel background model ignores information from the neighbourhood of a pixel, leaving it susceptible to locally occurring noise and camouflage. Additionally, Gibbs sampling introduces a certain amount of noise, meaning that the boundary between foreground and background is subject to a dithering effect. To resolve these issues a Markov random field is constructed, with a
node for each pixel, connected using a 4-way neighbour-
hood. It is a binary labelling problem, where each pixel
either belongs to the foreground or to the background.

The task is to select the most probable solution, where
the probability can be separated into two terms. Firstly,
each pixel has a probability of belonging to the back-
ground or foreground (data term), directly obtained from
the model as \( P(\text{bg}|x) \) and \( 1 - P(\text{bg}|x) \), respectively.
Secondly, there is the probability of being the same (smoothing term), which indicates how likely adjacent
pixels have the same assignment,

\[
P(l_a = l_b) = \frac{h}{h + m \cdot d(a, b)}, \tag{13}
\]

where \( l_y \) is the label of pixel \( y \), \( h \) is a parameter that sets
the half life, i.e. the distance at which the probability
becomes 0.5, and \( d(a, b) \) is the Euclidean distance be-
tween the two pixels in colour space. Noise suppression
is greatly improved if at least one of the neighbours
of a pixel have a high probability of being assigned the
same label - this is the purpose of \( m \). \( m \) is typically
1, but is decreased if a pixel is sufficiently far from its
neighbours that none provides a \( P(l(a) = l(b)) \) value
above a threshold. The decrease is such that at least one
of them matches the threshold.

Various methods can be considered for solving this
model. Graph cuts \[38\] would give the MAP solution,
however we use loopy belief propagation instead \[30\],
as it runs in constant time given an iteration cap, which
is important for a real time implementation; it also
runs considerably better on a GPU. A red-black \[30\]
update schedule is used to save memory - a checker-
board pattern is assigned to the pixels and then all
pixels of one colour send their messages, followed by all
pixels of the other colour, and so on, until convergence.
This results in half the memory consumption, as only
messages going in one direction need to be stored; it
additionally accelerates convergence \[30\]. Furthermore,
a hierarchical implementation is used, whereby the model
is first solved at a low resolution, before being scaled
up, such that the transferred messages accelerate conser-
gence at the higher resolution. This occurs over multiple
levels - the number is decided by halving the resolution
of the frame until either dimension drops below 32.

2.3 GPU Implementation

A background subtraction algorithm will have limited
practical use if it cannot be run in real time on a video
feed as it arrives. Accordingly, a GPU based implemen-
tation has been constructed. Because we have selected
a naturally parallel structure for the presented approach
adapting it to run on a GPU is relatively straightforward
- the model for each pixel is independent, whilst regular-
isation with belief propagation can be implemented
via message passing. The following phases exist:

1) Calculate the probability of the current pixel being
drawn from each mixture component in the model,
and also the new component probability. These
computations are all independent.

2) Pull the probabilities together, to generate the
probability of each pixel coming from the current
model. Update the model using Gibbs sampling
- another 19% is expended on colour conversion whilst
25% is spent estimating the lighting change, with the
remainder on input/output. Neither colour conversion
nor lighting change estimation were run on the GPU,
and all processing occurs in sequence without any par-
allelism between the CPU and GPU.

3) Calculate the costs for belief propagation - this is
naturally parallel.

4) Do message passing, over multiple hierarchical lev-
els. The use of a red-black update schedule means
that we can update half the pixels at each given
step.

5) Extract the final mask - an independent calculation
for each pixel.

Some approximations are made in the calculations to
accelerate run time - these are detailed in the supple-
mentary material.

Using the GPU implementation with a GeForce GTX
580 on a standard PC platform with a 4.4Ghz CPU, it
can obtain 28.5 frames per second on input that has a
resolution of \( 320 \times 240 \) (Calculated for the problem cd-
baseline-highway - see Section \[3\]). Note that only 41% of
the time is spent doing the background subtraction
another 19% is expended on colour conversion whilst
25% is spent estimating the lighting change, with the
remainder on input/output. Neither colour conversa-
nor lighting change estimation were run on the GPU,
and all processing occurs in sequence without any par-
allelism between the CPU and GPU.

2.4 Further Details

The core algorithmic details have now been given, but
other pertinent details remain.

The prior: The background model includes a prior on
the Gaussian associated with each mixture component.
Instead of treating this as a parameter to be set, it is
learnt from the data. Specifically, the mean and standard
deviation (SD) of the prior are matched with the mean
and SD of the pixels in the current frame, with the weight
so it is equivalent to a single sample,

\[
n_{i,0} = k_{i,0} = 1, \quad \mu_{i,0} = \frac{1}{|F|} \sum_{x \in F} x_i, \tag{14}
\]

\[
s^2_{i,0} = \frac{1}{|F|} \sum_{x \in F} (x_i - \mu_{i,0})^2, \tag{15}
\]

where \( F \) is the set of pixels in the current frame. To
change the prior between frames the posterior param-
eters must not be stored directly, instead offsets from
the prior are stored, which are then adjusted after each
update such that the model is equivalent.

Illumination change: The above helps by updating the
prior, but it does nothing to update the evidence. To
update the evidence a multiplicative model is used,
Camera Shake: With few exceptions [42, 43] background subtraction algorithms assume a fixed camera. In practice even fixed cameras shake due to vibrations in their mounting structure, which can be caused, for instance, by the wind or passing traffic. To improve vibration handling when updating the model we use a region around each pixel, rather than the pixel value directly. Going back to Equation 7 it updates the model assuming a single point value - it can be modified to accept a Gaussian distribution instead, to get

\[ \sigma^2_{i,m+1} = \sigma^2_{i,m} + w \varsigma^2_{i,m} + \frac{k_{i,m} w}{k_{i,m} + w} (x_i - \mu_{i,m})^2. \]  

where \( \varsigma \) is the standard deviation of the measured pixel value. Effectively it can be interpreted as the colour variance in the region around the pixel, from which we can randomly sample due to camera shake.

Assuming linear interpolation between adjacent values allows us to calculate the standard deviation of an image region, which we do for a single \( 1 \times 1 \) pixel region around the point sampled. If we treat the image surface as being made of square regions rather than triangular regions we only need the values of three corners for each square, which halves the number of pixels that need to be fetched. Using Figure 3 the region being calculated is given by the black square, which is split into four sub-squares. Three corners of the top right sub-square are given by \( x, h_n = \frac{x + x}{2} \) and \( h_e = \frac{x + x}{2} \); the same pattern applies to the other three. The standard deviation of these four squares is hence given by

\[
\varsigma = \frac{x^2 + h_n^2 + h_e^2 + h_w^2}{6} + \frac{h_n h_e + h_e h_w + h_w h_n}{8} + \frac{x h_n + x h_e + x h_w + x h_w}{12} - \mu^2
\]

(17)

where \( \mu \) is the mean of the four squares, given by \( \mu = \frac{h_n + h_e + h_w + h_w}{4} \). This has to be done for each of the three colour channels. As a further detail we scale the standard deviation with a parameter - a value of zero indicates that there is no camera shake, whilst 1 indicates a camera shake of about 1 pixel across. This approach will not scale for large amount of shake, but proves effective for the small quantities introduced by vibrations from, for instance, passing traffic.

Bootstrapping: The algorithm learns continuously - it does not need a learning period and can update as the scene changes. However, during initialisation the algorithm is not confident, and assumes everything belongs to the background. This makes the algorithm appear to be bad at bootstrapping - to obtain good performance it is forced to be over-confident. This is the opposite of confidence capping, in that it increases the confidence to the cap by multiplication. It is only applied whilst calculating \( P(bg|x) \) however - the actual parameters are not updated. This bias term exists purely to make the algorithm overconfident during initial model training, because some of the tests in Section 3 require results immediately from the first frame. For real world usage this would be switched off.

3 Experiments

Four data sets are used for the experiments:

- \textit{sabs} from Brutzer et al. [2]. This one is synthetic.
- \textit{wallflower} from Toyama et al. [1]
- \textit{star} from Li et al. [12]
- \textit{change detection (cd)} from Goyette et al. [13]. This has an online chart which is actively updated.

We perform extensive experiments using these data sets, allowing us to compare against a large number of alternative approaches. Brief summaries of many of the

4. Online at \url{http://www.vis.uni-stuttgart.de/index.php?id=sabs}
5. Can be downloaded from \url{http://research.microsoft.com/en-us/um/people/jckrumm/WallFlower/testimages.htm}
6. \url{http://perception.i2r.a-star.edu.sg/bk_model/bk_index.html}
7. Data set and chart online at \url{http://www.changedetection.net}
TABLE 1

Descriptions of the parameters for the DP-GMM. The final column contains the values used for the cd data set.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Description</th>
<th>Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>confidence cap</td>
<td>Equivalent to the learning rate in other approaches.</td>
<td>128</td>
</tr>
<tr>
<td>concentration</td>
<td>Concentration parameter for the DP - controls new mixture component creation.</td>
<td>0.01</td>
</tr>
<tr>
<td>min weight</td>
<td>Minimum weight for mixing in new samples, to accelerate learning time when the background changes.</td>
<td>0.0001</td>
</tr>
<tr>
<td>sd mult</td>
<td>Size of blurring kernel used to negate light camera shake.</td>
<td>0.6</td>
</tr>
<tr>
<td>component cap</td>
<td>Maximum number of components for each pixel.</td>
<td>8</td>
</tr>
</tbody>
</table>

approaches we compare against can be found in Table 1 in the supplementary material. Results for the wallflower data set are also available in the supplementary material, as are parameter sweeps to demonstrate that performance remains high regardless of parameters. Table 1 gives the approaches parameters.

3.1 SABS

Brutzer et al. [2] presented a synthetic evaluation of background subtraction algorithms, consisting of a 3D rendering of a road junction, traversed by both cars and people - see Figure 5. Despite being synthetic it simulates, fairly accurately, nine real world problems, and has the advantage of having ground truth for all frames. The nine real world problems are

- **basic**, a baseline test.
- **dynamic background**, which crops the area of analysis to be the area of a waving tree.
- **bootstrap**, which has no training period, such that a clean background plate is never seen.
- **darkening**, which simulates the sun setting.
- **light switch**, which has the street at night and a shop switching its light off then on again.
- **noisy night**, which is the scene at night with heavy noise.
- **camouflage**, where the cars and people have been coloured similarly to the background, so they blend in.
- **no camouflage**, same as camouflage, but with easy to see colours, for comparison.
- **H264**, which compresses the video heavily, to generate typical compression artefacts.

The f-measure is reported for the various approaches in Table 2 and is defined as the harmonic mean of the recall and precision.

\[
\text{recall} = \frac{tp}{tp + fn}, \quad \text{precision} = \frac{tp}{tp + fp} \quad (18)
\]

\[
\text{f-measure} = 2 \frac{\text{recall} \cdot \text{precision}}{\text{recall} + \text{precision}} \quad (19)
\]

using \( fp \) as the number of false positives, \( tn \) as the number of true negatives etc.

For this test we used one set of parameters for all scenarios, rather than tuning per problem. As can be seen, the presented approach takes first place for all scenarios, and is on average 27% better than its nearest competitor. In doing so it demonstrates that it is not sensitive to the parameters chosen, as it can produce top results in many scenarios without per-problem tuning. Running without regularisation is also included in the chart (denoted as DP, no post) - in all cases a lack of regularisation does not undermine its significant lead over the competition, demonstrating that the DP-GMM is doing most of the work, but that regularisation always improves the score, on average by 13%. It can be noted that the largest performance gaps between regularisation being off and being on appears for the noisiest inputs, e.g. noisy night, light switch, darkening and h264. These are the kinds of problems encountered in typical surveillance applications.

As a further point of comparison DP, con com is included, where our MRF-based post-processing has been swapped for the connected components method of Stauffer & Grimson [17]. Interestingly for the simpler problems it does very well, sometimes better than the presented method, but when it comes to the trickier scenarios the presented is clearly better. Figure 4 shows all the variants for a frame from noisy night. This demonstrates the key advantage of our post-processor - given a weak detection that falls below the implicit threshold it can convert it into a complete object, by
utilising both colour consistency and model uncertainty.

The frame shown in Figure 5 has been chosen to demonstrate two areas in which the algorithm performs poorly. Specifically, its robustness to shadows is not very effective - whilst this could be improved by reducing the importance of luminance in the colour space this has the effect of reducing its overall ability to distinguish between colours, and undermines performance elsewhere. The second issue can be seen in the small blobs at the top of the image - they are actually the reflections of cars and people in the scene. Using a DP-GMM allows it to learn a very precise model, so much so that it can detect the slight deviation caused by a reflection, when it would be preferable to ignore it. Further processing could potentially avoid this.

3.2 Star

The star data set [12] is very similar to the wallflower data set - bootstrap from wallflower is the same video as br in this data set. To its advantage star has an improved testing procedure however, as it provides multiple test frames per problem, with performance measured using the average similarity score for all test frames, where similarity $= tp/(tp + fn + fp)$. In addition the sequences are generally much harder, due to text overlays, systemic noise and some camera shake. Fewer algorithms have been run on it. The presented approach takes 1st 7 times out of 9, beaten twice by Maddalena et al. [19], though in both cases by a small margin. The light switch test in this data set does not trip it up this time - the library where it occurs has a high ceiling and diffuse lighting, making multiplicative lighting much more reasonable. Complex dynamic backgrounds clearly demonstrate the strength of a DP-GMM, as evidenced by its three largest improvements (cam, ft and ws).

3.3 Change Detection (cd)

The change detection [13] data set is much larger than any of the others, and includes a dense ground truth - masks are provided for all frames past some initial training period. It additionally limits parameter tuning such that a single parameter set must be used for all 31 videos. Video resolution is not great however, and many of the videos are of dubious quality; many appear to have been post-processed, often with a low quality de-interlacing algorithm that leaves ghosts. The videos are divided into six categories, which can be seen alongside the results in Figure 7. Quantitative results can be found in Table 4. The online version considers many different scoring mechanism, and then combines them in a non-linear rank based system. Instead we present the f-measure scores only, as it is the most used metric.

We now consider each test in turn.

- **Baseline.** A basic set of videos, with nothing unusual, though pedestrians has saturated colour. All the top algorithms get comparable scores for this test.

10. We tune per-problem, as all the competitors have done the same. Results for a single set of parameters are also presented.

11. From Figure 7 the traffic video demonstrates ghosting; the abandoned box video is an example of another kind of defect. It should be noticed that the other real data sets have similar issues.

12. The table is accurate as of February 2013 - an up to date version can be found at http://changedetection.net

13. This is problematic - the addition of a new approach can cause two unrelated algorithms to switch places. This can even occur when the new algorithm is much better or worse than those it reorder.
Fig. 6. Star results: On the top row is the image, on the second row the ground truth and on the third row the output of the presented algorithm. We use the same frames as Culibrk et al. [44] and Maddalena & Petrosino [19], for a qualitative comparison. The videos are named using abbreviations of their locations.

<table>
<thead>
<tr>
<th>method</th>
<th>cam</th>
<th>ft</th>
<th>ws</th>
<th>mr</th>
<th>lb</th>
<th>sc</th>
<th>ap</th>
<th>br</th>
<th>ss</th>
<th>mean</th>
</tr>
</thead>
<tbody>
<tr>
<td>Li 2 [12]</td>
<td>1.1596 (5)</td>
<td>0.9999 (6)</td>
<td>0.9667 (6)</td>
<td>1.841 (6)</td>
<td>1.554 (6)</td>
<td>0.5209 (6)</td>
<td>1.113 (6)</td>
<td>0.3079 (6)</td>
<td>1.294 (6)</td>
<td>1.193 (6)</td>
</tr>
<tr>
<td>Stauffer [17]</td>
<td>0.0757 (6)</td>
<td>0.6854 (3)</td>
<td>0.7948 (4)</td>
<td>0.7500 (5)</td>
<td>0.6276 (4)</td>
<td>0.5694 (4)</td>
<td>0.3923 (4)</td>
<td>0.4779 (4)</td>
<td>0.4928 (4)</td>
<td>0.5600 (4)</td>
</tr>
<tr>
<td>Culibrk [44]</td>
<td>0.5236 (4)</td>
<td>0.4636 (5)</td>
<td>0.7500 (5)</td>
<td>0.7368 (5)</td>
<td>0.6276 (4)</td>
<td>0.5694 (4)</td>
<td>0.3923 (4)</td>
<td>0.4779 (4)</td>
<td>0.4928 (4)</td>
<td>0.5600 (4)</td>
</tr>
<tr>
<td>Maddalena [19]</td>
<td>0.6960 (3)</td>
<td>0.6544 (4)</td>
<td>0.8247 (5)</td>
<td>0.8247 (5)</td>
<td>0.6276 (4)</td>
<td>0.5694 (4)</td>
<td>0.3923 (4)</td>
<td>0.4779 (4)</td>
<td>0.4928 (4)</td>
<td>0.5600 (4)</td>
</tr>
<tr>
<td>DP-GMM</td>
<td>0.3567 (2)</td>
<td>0.7049 (2)</td>
<td>0.9090 (2)</td>
<td>0.9090 (2)</td>
<td>0.6276 (4)</td>
<td>0.5694 (4)</td>
<td>0.3923 (4)</td>
<td>0.4779 (4)</td>
<td>0.4928 (4)</td>
<td>0.5600 (4)</td>
</tr>
<tr>
<td>DP-GMM, tuned</td>
<td>0.7876 (1)</td>
<td>0.7424 (1)</td>
<td>0.9298 (1)</td>
<td>0.8411 (1)</td>
<td>0.6665 (1)</td>
<td>0.6733 (1)</td>
<td>0.5675 (2)</td>
<td>0.6496 (1)</td>
<td>0.5522 (2)</td>
<td>0.7122 (1)</td>
</tr>
</tbody>
</table>

TABLE 3

Star [12], [19] results: Refer to Figure 6 for exemplar frames, noting that lb has abrupt lighting changes. The average improvement of DP, tuned over its nearest competitor is 5%.

$$\text{TABLE 4}$$

CD [13] results: This chart only contains the most competitive algorithms - the online chart includes many more. The rank after each f-measure takes into account the omitted approaches. Refer to Figure 7 for example frames.

- **Dynamic Background.** These videos have complex multi-modal backgrounds, such as water. As demonstrated previously, the presented excels at such input, outperforming the competitors significantly.
- **Camera Jitter.** The camera is not properly mounted for these videos, and as such shakes. This does reduce the quality of our output - the output frames tend to miss parts of objects due to the wider background model induced by the shaking. It also incorrectly classifies the background during the worst shakes, but we still manage a close second place.
- **Intermittent Motion.** This involves objects being left alone (static), and is our weakest result. Other approaches often have a specific code path to handle this situation, whilst we do not, putting us at a disadvantage.
- **Shadow.** These videos have shadows, and test the ability of the approach to ignore them. In our case we have some capability to handle soft shadows, as seen in cubicle, but will typically detect hard shadow regions. Interestingly our result remains among the top approaches despite its limited capabilities.
- **Thermal.** A somewhat unusual video source, this demonstrates cameras for low light recording. Such sources have a large amount of noise, which the presented is expected to do well on. Unsurprisingly it demonstrates a strong lead over the competition. It actually suffers from being too sensitive - in corridor it detects the thermal reflections when these...
Fig. 7. CD results: Layout is identical to Figure 6, with multiple rows. The ground truth includes various shades of grey - a dark grey to indicate shadowed regions, a mid grey for areas that are ignored when scoring for all frames and a light grey for areas ignored for just this frame. Mid grey is used to focus on the action being tested, whilst light grey indicates areas where foreground/background assignment is ambiguous (typically presents as a thin line around each segment).
should be considered background.
Overall we win on average for the cd data set (Table 4). This is because the DP-GMM model can handle backgrounds that are complex and dynamic, due to its ability to estimate mixture components automatically and avoid over-/under-fitting. This ability, in combination with the regularisation based on a Markov random field allows it to effectively cope with heavy noise.

3.4 Mixture Count
The approach implicitly calculates a mixture count for each pixel\[14\]. Figure 8 visualises this. (a) - (d) show two frames from the star data set, and their corresponding per pixel mixture counts for the thousandth frame. The images are coloured such that black is 0 components and white is 4 - the majority of pixels in both frames have been estimated to have 1 component. Trees moving in cam create components where the movement is large, though note that most areas can be handled using a single wide variance component. In ss the escalators again generate extra components, though being mostly black they can often be covered by a single component. The heavily used escalator on the right gets lots of components because the approach forms one for the people going past, in case they stop and become part of the background. These components are never considered to be part of the background as they don’t obtain enough weight. (e) demonstrates the mixture count changing over time, with a synthetic video. The video shows 10 seconds of 1 mode, then 2 modes, and so on up to 4, then back again, at 24 fps - the ground truth mode count is plotted\[15\] . We can plot the mode count, averaged over all the pixels - it follows ground truth, with lag as expected. Note that the lag is very short when learning a new mixture component - only about 2 seconds, but very slow when it comes to forgetting a background component, which is desirable.

4 Conclusions
We have presented a new background subtraction method and validated its effectiveness with extensive testing. The method is based on an existing model, namely DP-GMMs, but with new model learning algorithms developed to make it both suitable for background modelling, and computationally scalable. Whilst the basic concept of per-pixel density estimate (DE) can be traced back to Stauffer & Grimson\[17\], the use of a non-parametric Bayesian model gives it a significant advantage when it comes to handling dynamic backgrounds. Specifically, our model is able to learn the number of mixture components more accurately and hence better cope with a variety of scene changes, in comparison with the existing more heuristic DE methods. It also proves itself to have good performance in many other areas, particularly on dealing with heavy noise. Despite its thorough theoretical basis implementation remains relatively simple\[16\].

A number of improvements can be considered. Combining information between pixels only as a regularisation step does not fully exploit the information available. A more rigorous method of spatial information transmission would be desirable - a dependent Dirichlet process could provide this. Sudden complex lighting changes are not handled, which means it fails to handle some indoor lighting changes. Furthermore, a more sophisticated model of the foreground and an explicit model of left objects could further improve our method. Solutions to these problems from many of the existing methods could be adapted to ours\[6\], \[33\], \[35\].

References

15. The video changes colour repeatedly, cycling though all modes every second and showing each for an equal amount of time. It includes noise.
16. 186 lines of C for the DP model and 239 lines for the post-processing. The GPU implementation comes in at 814 lines, due to the complex processing structure needed for an efficient implementation.


