This section discusses the implementation of the moving object detection algorithms. It will examine the necessary modifications to support HE video data in §\ref{sec:adaptations}, and detail the more complex algorithms needed for unsupervised learning in §\ref{sec:OMM} and §\ref{sec:EMAlg}.

# Homomorphic Encryption Adaptations

There were two main challenges to overcome when implementing inference algorithms for HE. Firstly, the number of operations that can be applied is limited by the depth of the ciphertext. Secondly, the set of operations supported by HE is more limited than is available for plain data. Consequently, compromises were made to produce accurate results without introducing detrimental side effects – for example, to accommodate more operations, the depth of a ciphertext could be increased, but this significantly increases the size of ciphertexts, making data transmission infeasible.

The adaptations implemented for the less complex algorithms are detailed below. The techniques investigated for implementing GMMs are evaluated in §\ref{sec:integration}.

## Frame Differencing

Frame differencing required few modifications to accommodate HE data. Only a single operation is required: \textit{subtraction}, and is provided by the standard CKKS implementation. Moreover, subtraction does not impact a ciphertext’s level, so does not require size modifications. As well as making networking more efficient, this makes the inference algorithm faster because the data being operated over is minimised.

Therefore, the only modification required to support HE data is to replace the subtraction function in the traditional algorithm with the subtraction circuit of a HE library.

## Mean Filter

A naïve mean filter implementation recalculates the reference frame repeatedly by storing a list of all observed frames and computing their mean each when a new frame is received. Alternatively, a more advanced implementation will use an iterative function to update the mean, removing the requirement to store preceding frames.

Evidently, the second method will produce better time and space complexity when considering plain video data. However, when investigating HE data, the distinction is not as clear. The second method becomes problematic in that the mean must have both \textit{multiplication} and \textit{addition} operations applied when updating. Each time a multiplication circuit is applied, the ciphertext will be reduced. Therefore, the ciphertext must have the same number of levels as frames in the video. This quickly becomes infeasible; increasing ciphertext size so far would severely increase the time complexity of operations and make transmission times impractically slow. Consequently, this method can be immediately ruled out.

The first method encounters different difficulties. One issue is the space complexity of storing all frames in the video. Since HE data is very large, storing multiple copies of each frame significantly increases memory usage. Similarly, HE operations are noticeably slower than plaintext operations. Therefore, while the cost of recalculating the mean of plaintext data may be negligible, HE implementations will become increasingly slower. However, a solution can be derived. A compromise can be achieved by limiting the number of frames stored, forgetting the oldest frame whenever a new one arrives. Importantly, this may reduce inference accuracy, so a balance must be struck between running time and inference quality.

## Gaussian Average

Similarly to a mean filter, a plaintext Gaussian average implementation might fit distributions using iterative formulae for the mean and variance. Copied from §\ref{sec:gaussianAverage} for convenience, Equation \ref{eq:mean2} and Equation \ref{eq:variance2} provide these formulae. Consequently, the plaintext implementation will be infeasible in the HE domain. Fortunately, adaptations can be made.  
EQUATION

where $\alpha$ determines the temporal window, $d$ represents the Euclidean distance between a pixel and the mean, and $c$ is some constant.

Firstly, expanding the iterative formulae highlights that the frames of a video are \textit{multiplicatively independent}, as demonstrated by Equation \ref{eq:expansion} for the fifth frame of a video.

EQUATION

Helpfully, the coefficient terms across calculations are identical, so computation can be shared. The server decides the value of $\alpha$ before runtime, so coefficients can be pre-calculated in the clear before being encoded for HE multiplication. This means that only a single multiplication needs to be applied to each frame. Then, results can be summed to give the mean and variance. Consequently, the number of levels required for a ciphertext is significantly reduced, so time and space complexity are also reduced.

However, this method still requires storing a list of preceding frames to enable recalculating terms. Like with mean filter, this introduces a trade-off between running time and memory usage against inference accuracy.

# Online Mixture Model

In 1999 Stauffer and Grimson proposed \textit{adaptive background mixture models} for real-time moving object detection [STAUFFER]. To overcome a single Gaussian distribution’s inability to model natural pixel variation, they proposed a mixture of adaptive Gaussians. The advantage of their technique over other GMMs is that the model runs \textit{online}: the model can be fitted, and results returned in a single phase. While this is useful for real-time inference acting on a constant data stream, it has the disadvantage of producing less accurate results earlier in the execution sequence.

## Fitting

For a particular pixel, the values that it records are known as the \textit{pixel process}. This is a time series of pixel values such that, at any time $t$, the process of pixel $(x,y)$ is defined by Equation \ref{eq:pixelProcess}.

EQUATION

where $I$ is the image sequence.

The history of a pixel can be modelled as a mixture of $K$ Gaussian distributions. $K$ is usually between 3 and 5, depending on available memory and computational power availability. Given a pixel process, the probability of observing the pixel value at time $t$ is given by

EQUATION

where $\omega\_{i,t}$ represents an estimate of the proportion of the data accounted for by the $i^{\text{th}}$ Gaussian at time $t$, $\mu\_{i,t}$ and $\Sigma\_{i,t}$ are the mean and covariance matrix of the $i^{\text{th}}$ Gaussian at time $t$ respectively. $\eta$ is the Gaussian probability density function given by Equation \ref{eq:pdf}.

To maximise the likelihood of the observed data, a \textit{k-means} approximation was selected to engender an online implementation. Each pixel in a new frame is compared against the existing Gaussian distributions until a \textit{match} is found. A match occurs when a pixel value is within a predefined number of standard deviations of a distribution. The number of standard deviations will vary across distributions as each distribution will account for different factors such as lighter or shadier regions.

If none of the distributions match a pixel’s value, the least likely Gaussian is replaced by a new distribution defined with the pixel as its mean, an initially high variance, and low prior weight. Then, the prior weights are adjusted at time $t$ using Equation \ref{eq:priors}.

EQUATION

where $\alpha$ is a learning rate, and $M$ is an indicator function of $1$ if Gaussian $k$ at time $t$ matched, and $0$ otherwise. After this approximation is complete, the weights are normalised.

For unmatched distributions, $\mu$ and $\sigma$ are unchanged. The parameters of the matching distributions are updated according to Equation \ref{eq:muAndSigma}, where $rho$ is defined by Equation \ref{eq:rho}.

EQUATIONS

## Predicting

Once the parameters have been updated, each pixel is labelled by determining the Gaussian component most likely produced by the background process. This decision assumes that there will be little variance in the Gaussians when the frame is static and a large variance when a new object occludes the background. Consequently, a function defining the proportion of the GMM representing the background process is required.

First, the Gaussians are ordered by the value of $\frac{\omega}{\Sigma}$. The definitions of $\omega$ and $\Sigma$ mean that this value will increase as the distribution gains more evidence and the variance decreases. This value will only differ from the last iteration for matching distributions, so the sorting process can be made more efficient. The ordered list can then be iterated over, and the first $B$ distributions are taken as the \textit{background model}, where

EQUATION

The threshold, $T$, is a measure of how much data should be accounted for. In other words, the best-fitted distributions are taken until a certain portion of recent data has been considered.

Once the background model has been decided, it can be used to label the pixel as either \textit{foreground} or \textit{background}, allowing the moving objects to be extracted as foreground.

# Expectation-Maximisation Algorithm

Proposed by Dempster et al.\ in 1977, the \textit{expectation-maximisation} (EM) algorithm is a general iterative method for maximising the likelihood of \textit{latent variables} in statistical models [DEMPSTER]. The algorithm contains two stages: the expectation stage, or \textit{E-step}, and the maximisation stage, or \textit{M-step}, which are iterated over until the model converges. The E-step generates a function for the expectation of the likelihood of data points occurring given the current model parameters. The M-step computes new parameters to maximise the function found in the E-step. While this will always increase the \textit{marginal likelihood function}, the EM algorithm does not guarantee convergence on a maximum likelihood estimator, but rather a local maximum. To overcome this, techniques such as \textit{random-restart hill climbing} can be employed [HILLCLIMBING].

Focussing on GMMs, the algorithm can assign observed data points to components of the model such that the likelihood of the components generating the points is maximised. The below process can formalise the E-step. First, the \textit{pseudo-posterior} – the probability that an observation, $X\_i$ belongs to a component $Z\_k$ - is calculated using Equation \ref{eq:eStep1}.

EQUATION

where $\omega\_k$ is the component weights of component $k$ and $\mathcal{N}(x\_i, \mu\_i, \sigma\_i)$ gives the probability of $x\_i$ under component $k$.

The \textit{auxillary function} defined by Equation \ref{eq:eStep2} can then be applied to, $\gamma\_{Z\_i = k}$, where $\theta\_{t-1}$ is the parameter generated in the previous iteration and $\theta\_t$ is the new parameter value. Using Jensen’s inequality, it can be proven that this auxiliary function is the lower bound of the gain of the likelihood that is obtained by updating the parameter values.

EQUATION

where $\log \mathbb{L} (\theta\_k \; | \; X, Z)$ is the log likelihood of a Gaussian component with updated parameters and $\probP(Z\_k | X, \theta\_{t-1})$ is the distribution of latent variables according to the current parameters.

After the auxiliary function has been generated, the M-step can begin. This means maximising the value of $Q$ to produce the optimal parameter value in Equation \ref{eq:mStep1}.

EQUATION

From this, the optimal parameter values can be derived by differentiating Equation \ref{eq:mStep2} with respect to the means, covariances, and weights, and solving when equal to zero. The results of these calculations are given Equation \ref{eq:mStep3}, Equation \ref{eq:mStep4}, and Equation \ref{eq:mStep5}, respectively, where $N\_k = \sum^N\_{i=1} \gamma\_{Z\_i = k}$.