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}. Adaptations were required to incorporate the HE Boolean circuits and overcome operation depth limitations.

# Homomorphic Encryption Adaptations

There were two main challenges to overcome when implementing inference algorithms in the HE domain. 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 when working with 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 memory usage of ciphertexts, so data transmission quickly becomes infeasible.

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

## Frame Differencing

Frame differencing is a relatively simple algorithm to adapt for the HE domain. It only requires a single operation, \textit{subtraction}, that is provided by the standard CKKS implementation. Moreover, subtraction does not impact the level of a ciphertext, so has no impact on ciphertext size. As well as making networking more efficient, this also makes the inference algorithm faster because the data being operated over is smaller.

Therefore, the only modification required to operate over HE data is to replace the subtraction function in the traditional algorithm with a call to the subtraction circuit provided by the HE library.

## Mean Filter

The most straightforward implementation recalculates the reference frame repeatedly by storing a list of all frames that have been observed and finding their mean each time a new frame is received. In contrast, a more advanced implementation will use an iterative function to update the mean, so does not require storing all frames.

Evidently, the second method will perform better in both 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 to it during the updating phase. 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 because increasing ciphertext size so far would severely increase the time complexity of operations and make transmission times impracticably slow. Consequently, this method can be immediately ruled out

The first method encounters different difficulties. One particular issue is the space complexity of storing all frames in the video. Since HE data can get very large, storing multiple copies of each frame significantly impacts the application's memory usage. Similarly, HE operations are noticeably slower than plaintext operations. Therefore, while the cost of recalculating the mean of plaintext data may be relatively negligible, as more frames are observed, HE implementations will become considerably slower. However, this method can be used to derive a solution. 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 implementation of Gaussian average inference would fit the Gaussian 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, it can be adapted.  
EQUATION

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

Firstly, it can be observed that 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 the video.

EQUATION

Helpfully, the coefficient terms across calculations are identical, so computation can be shared. More importantly, the server decides the value of $\alpha$ before runtime. Therefore, the 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 recalculate terms repeatedly. 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 cope with the changing lighting conditions in practice, they proposed a mixture of adaptive Gaussians. The advantage of this technique over other GMMs is that the model runs \textit{online} because no training phase is required. Instead, the model can be fitted, and results returned in a single phase. While this is useful for real-time inference acting on a constant stream of data, it has the disadvantage of producing less accurate results earlier in the execution sequence.

## Fitting

For a particular pixel, the values that occur over time 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 a value 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 the online aspect of the system. Each pixel in a new frame is compared against the existing $K$ 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 current value 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, the $\mu$ and $\sigma$ parameters are unchanged. However, 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 must be labelled by determining the Gaussian component most likely produced by the background process. This decision assumes that there will be relatively little variance in the Gaussians when a static object is in the frame and a large variance not matching existing distributions when a new object occludes the background. Consequently, a method defining the proportion of the GMM representing the background process is required.

To achieve this, first, the Gaussians are ordered based on 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 the 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} of a statistical model [DEMPSTER]. There are two stages in the algorithm: 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}, there is no guarantee that the EM algorithm will converge to a maximum likelihood estimator: the algorithm may converge on a local maximum. To overcome this, techniques such as \textit{random-restart hill climbing} can be employed [HILLCLIMBING].

Although the EM algorithm can be applied to any statistical model, this dissertation will discuss its application to GMMs. The algorithm can be used to assign observed data points to components of the model such that the likelihood of the components generating the points is maximised. When applied to a GMM, the E-step can be formalised by the below process. To begin with, 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 the result, $\gamma\_{Z\_i = k}$, 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, but this proof is excluded for brevity.

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, in turn. The results of these calculations are given Equation \ref{eq:mStep3}, Equation \ref{eq:mStep4}, and Equation \ref{eq:mStep5}, respectively. In the equations, $N\_k = \sum^N\_{i=1} \gamma\_{Z\_i = k}$.