Introduction to Multimedia

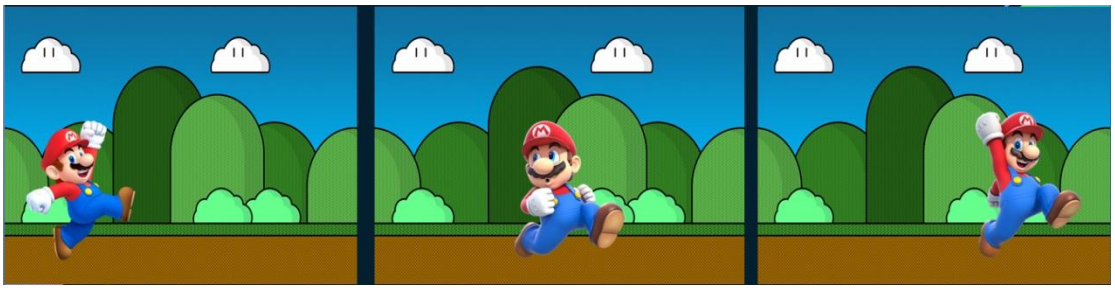
Background Subtraction

101033252 陳冠宏
101062331 周彥儒

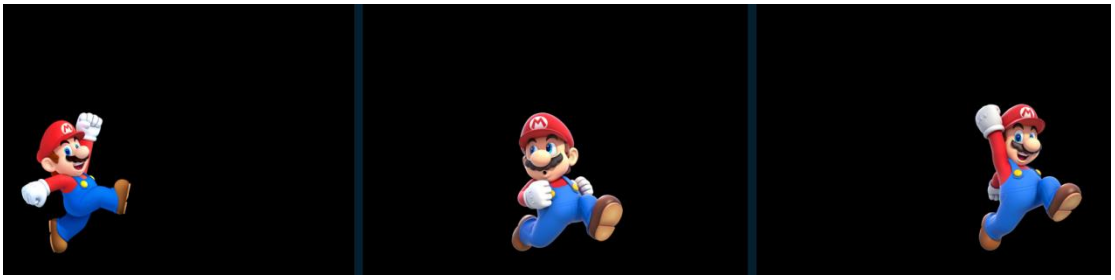
Problem Descriptions:

在固定靜止拍攝的視訊影片中進行前景物體的偵測，藉以分割出前景的物體與背景影像，此類的偵測技術可應用在各種不同的領域，包括醫學影像分析、背景重建、物件追蹤、交通監視系統、行人偵測等，下列以圖文說明之：

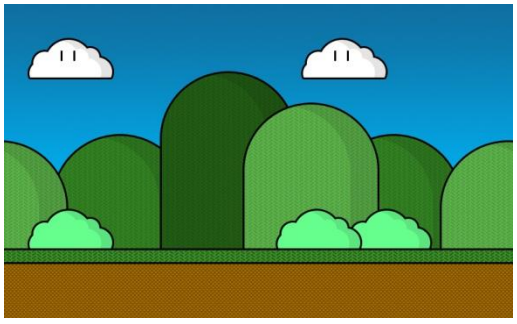
A sequence of video frames:



After background subtraction:



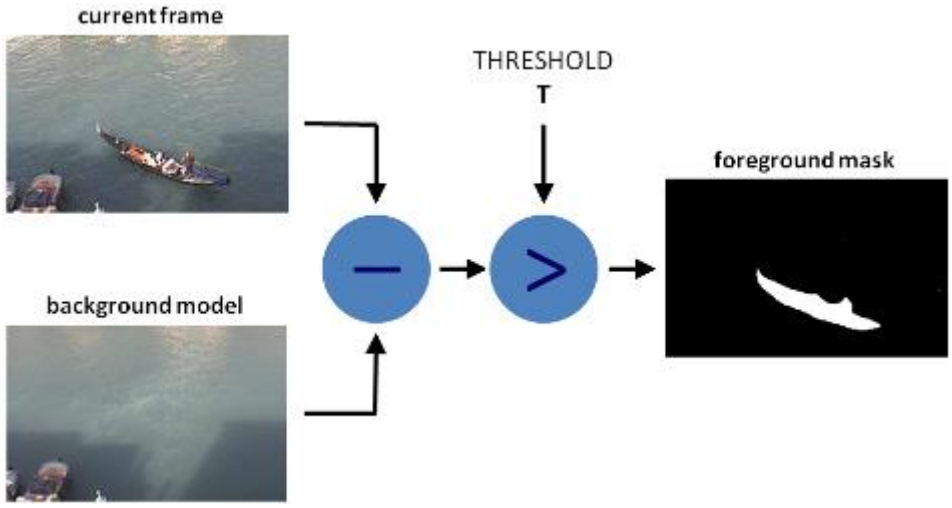
foreground objects ↓



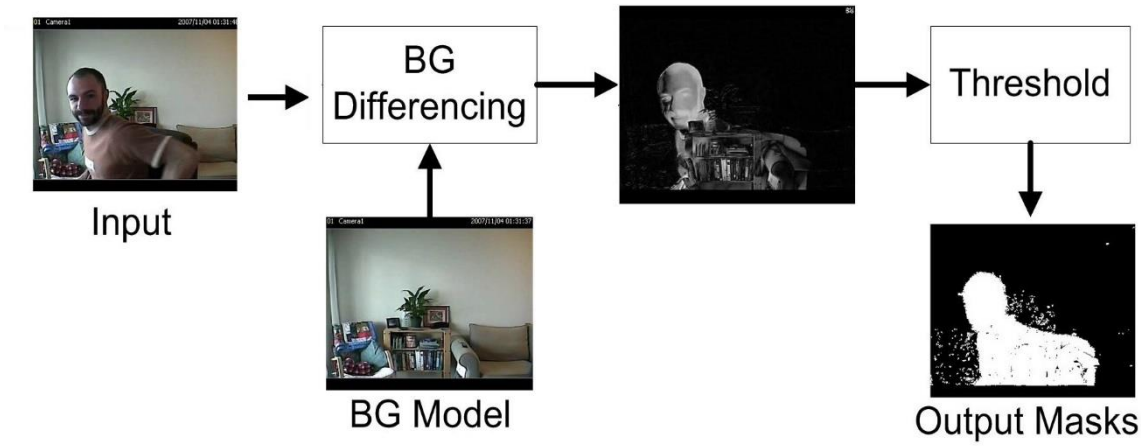
Background ↑

Problems Encountered:

影像或影片進行 background subtraction 最簡單的方式即為直接相減，並使用簡單的 threshold 判斷前後景，在理想狀況下，就可以得到一個分離的 foreground mask，如下圖所示：



然而，在一般情形下，場景會隨著時間而產生小部分的異動，如亮度、光影變化、物體細微的變動(風吹動的樹葉或草皮)、攝影機的輕微晃動等都是常見須解決的問題，即使是靜止的背景，pixel 值可能也會有些微改變。如下圖可以發現，Output Mask 有明顯的雜訊，無法精確分離前後景。



因此，更好的做法是使用一個優良並能時常更新背景影像的背景模型，分析顏色、像素及區塊資訊成為描述背景的特徵。此外，在複雜前景的移動干擾下，例如在人群、車流頻繁來往的拍攝場合，亦或是汽車長時間的背景停留，如何取得完整的街景或風景，也是一個困難的問題。

隨著以上問題規模的擴大，也大大增加了時間複雜度與其計算成本；因此，在「分割前後景的精密度」與「及時處裡」中，兩者取得良好平衡點亦是待解決的問題之一。

Algorithm:

單高斯模型(Gaussian Model)

高斯函數有非常多漂亮的性質，幫助我們在此影像處理、分離前後景的過程中，過濾不需要的雜訊，並建立一個良好的背景模型，並透過不斷更新此背景模型，得以更精確的分離出結果。

下列是一個高斯分布的機率密度函數：

均值為 μ ，變異數為 σ^2 (或標準差 σ)

$$f(x;\mu,\sigma) = \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x-\mu)^2}{2\sigma^2}\right)$$

透過以上函數，稍作改變成為多維的單高斯模型(Gaussian Model)，如下：

$$g(x;\mu,\Sigma) = \frac{1}{\sqrt{(2\pi)^d |\Sigma|}} \exp\left[-\frac{1}{2}(x-\mu)^T \Sigma^{-1}(x-\mu)\right]$$

其中 x 是維度為 d 的樣本向量， μ 是密度函數中心點，∑為共變異矩陣。

假設 $x_1, x_2 \ldots\ldots x_n$ 為獨立事件，則發生 $X = \{x_1, x_2 \ldots\ldots x_n\}$ 之機率密度為

$$p(X;\mu,\Sigma) = \prod_{i=1}^{n} g(x_i;\mu,\Sigma)$$

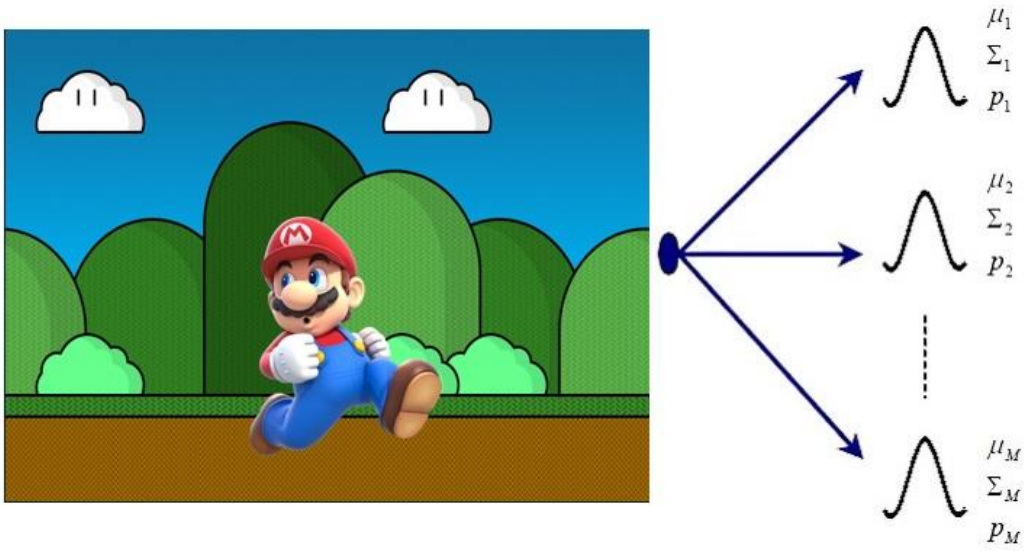
透過 Maximum Likelihood Estimation 找出 μ 與∑使其 p(X, μ ,∑)有最大值。

然而，在幾何空間的觀點上，一個單高斯模型的在二維空間的分布應該近似於橢圓，在三維空間上的分布應該近似於橢圓球型。然而現實，藉由特徵以判斷是否屬於同一類別的問題並不滿足「橢圓」分佈的特性，而單高斯模型僅能用一個平均值來代表一群樣本在空間上的中心點，並使用共變異矩陣來描述分布的形狀。

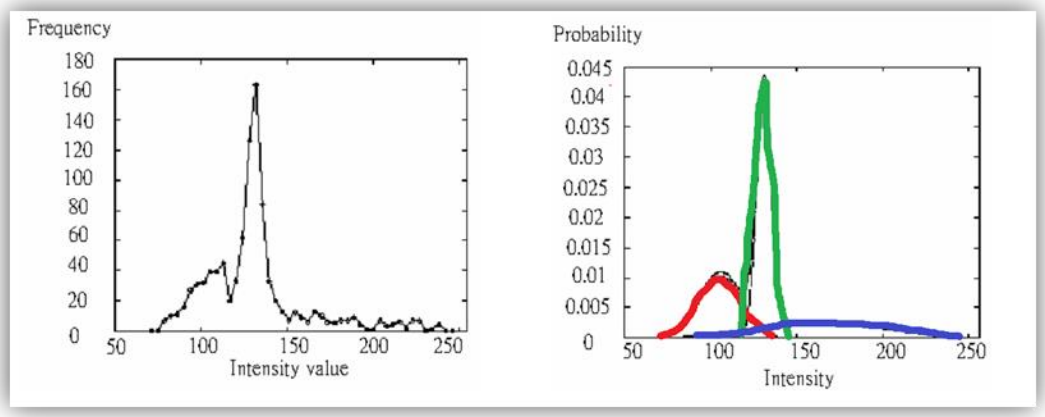
因此，最直觀的觀點，就是用多個高斯模型以描述多個特徵向量在空間上的分布，藉以得到更精確的結果。

高斯混合模型(Gaussian Mixture Model)

由前述得知，如果我們的樣本點在 d 維空間中不是橢圓球狀，使用單高斯模型的問題在於無法完整描述一張圖的特徵；因此，我們同時用多個高斯模型，而高斯混合模型透過增加 Model 的個數，我們可以任意地逼近任何連續機率密度函數分佈。



以簡單的觀點說明，我們將一個機率密度分布函數可以拆成 K 個高斯分布之加權平均，如下圖：



左邊的機率密度分布函數無法用單一一個高斯函數表示，但以此例我們可以用三個不同參數的高斯分布加權平均去逼近原機率密度分布函數，也就是高斯混合模型的核心思想。

假設我們的樣本資料$X_N = \{X_1, X_2, \ldots . X_n\}$ 在維度 d 空間中分布，則我們的高斯混合模型可以以下列式子表示：

$$p(x_N \mid \lambda) = \sum_{i=1}^{M} w_i g_i(x_N)$$

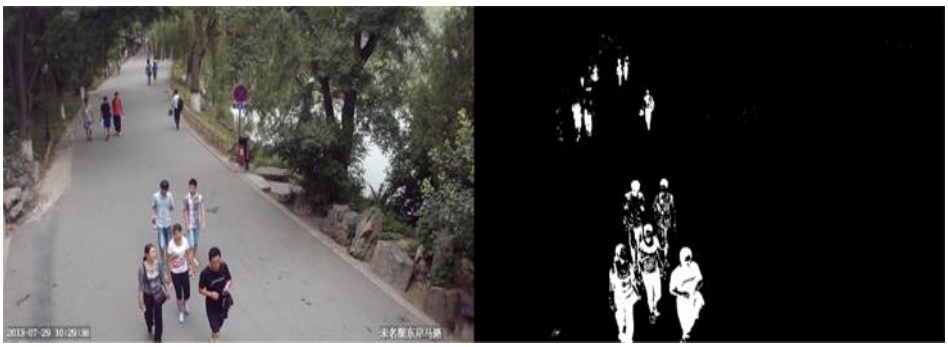
$$g_i(x) = \frac{1}{(2\pi)^{D/2} \mid \Sigma_i \mid^{1/2}} e^{D}$$

M 表高斯模型個數，w_i表第 i 個高斯模型之權重，g_i表第 i 個高斯模型之函數。

我們的目標是找到參數讓p($x_N|\lambda$)越大，
有了混合高斯模型的演算法之後，使用類似於 K-means cluster 方法，來初始化我們要的高斯模型，步驟如下：

1. 蒐集 N 個特徵向量，作為 training 的資料。
2. 初始化，隨機抽取 K 個做為 cluster 中心。
3. 對其他 N-K 個向量對 K 個 cluster 中心算出距離，再以最短距離作為 cluster 的依據，每個向量被分配到其對應的 cluster 中心。
4. 對每一個 cluster 算出向量平均值，再以此找出新的 cluster 中心。
5. 判斷是否收斂，如無收斂則重複步驟 3、4；如已收斂則往下步驟。
6. 判斷是否合併 cluster(太近)，如需合併，則重新做步驟 2。
7. 完成高斯混合模型的初始化參數，以每個 cluster 的資料個數、中心等資訊，當作各高斯模型的參數。

透過每張 frame，分析其特徵向量，不斷的學習並更新高斯混合模型，就可以精確判斷前後景。



觀察上圖，高斯混合模型可以降低雜訊的影響。

Improve the quality, accuracy, speed, or others:

參考（官方版本）：
http://docs.opencv.org/3.1.0/d1/dc5/tutorial_background_subtraction.html#gsc.tab=0

OpenCV 3 版本提供了 MOG2 這個強大的 API 給大家使用，讓我們可以快速簡單的實作出 background-subtraction。

不過官方的版本存在一個很大的缺點，就是並沒有特別針對 ”雜訊” 做處理。因為串流影像隨著時間的變化，光線亦會有變化，使用官方的 API 對於太過細緻的部分會很敏感，舉例來說，如果畫面上突然出現一些微小細節的變化（例如:光線變化、風吹草動），雖然對於人眼來說，可能並不會去理會，甚至根本沒有察覺到，但對於電腦來說，還是會將它視為前景並且分離出來，這些不應該被分離出來的部份，我們就把它稱作為 ”雜訊”。

因此，我們的改良方式著重於對每一張 frame 作出預先的處理，要解決雜訊的問題，我們想到的方法是將影像模糊化，如此一來，許多細微變化就會被忽略掉，這樣就可以減少雜訊對串流影像的影響。

利用 blur() 函式可以有效地處理這個問題

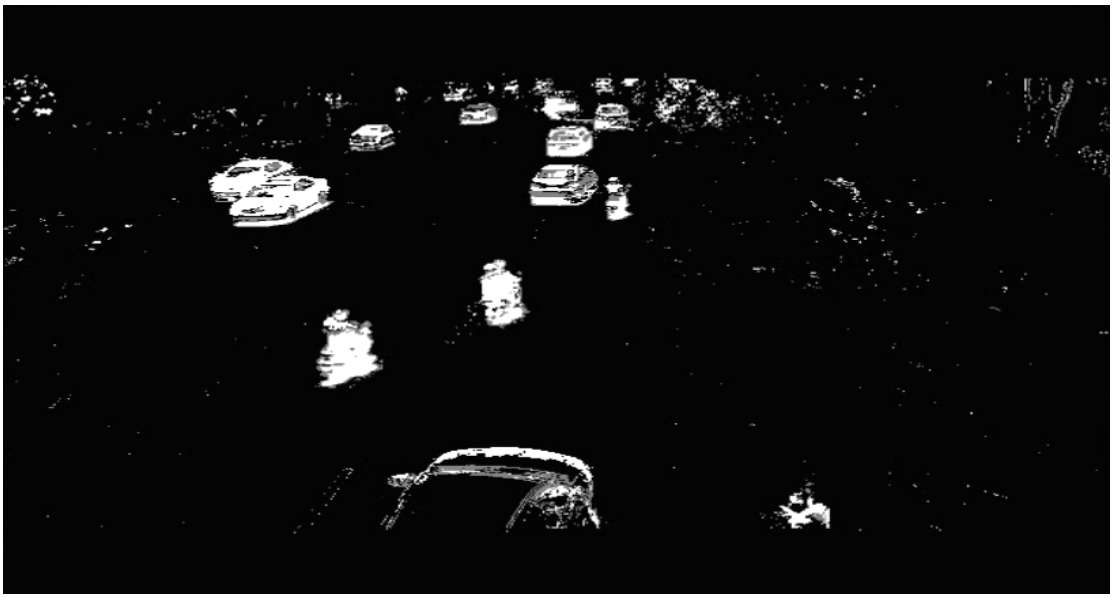
```
void cv::blur ( InputArray   src,
                OutputArray  dst,
                Size         ksize,
                Point        anchor = Point(-1,-1),
                int          borderType = BORDER_DEFAULT
              )
```

Blurs an image using the normalized box filter.

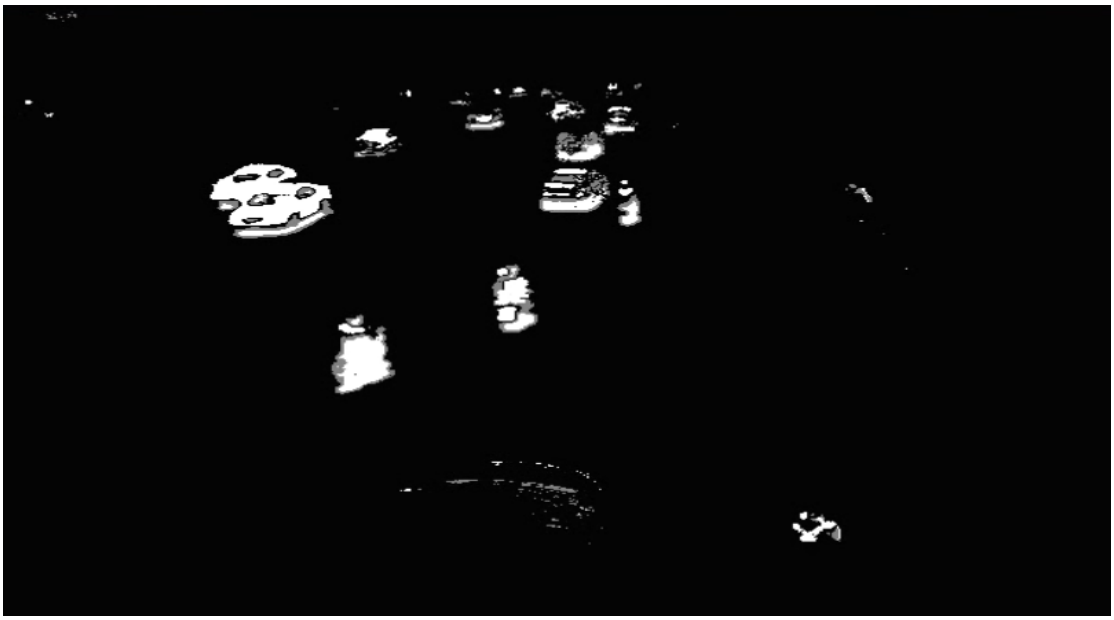
The function smoothes an image using the kernel:

$$K = \frac{1}{\texttt{ksize.width*ksize.height}} \begin{bmatrix} 1 & 1 & 1 & \cdots & 1 & 1 \\ 1 & 1 & 1 & \cdots & 1 & 1 \\ \hdots & & & & & \\ 1 & 1 & 1 & \cdots & 1 & 1 \end{bmatrix}$$

此外，我們還使用了 morphologyEx 進行開運算 (對前景影像漏洞有填補的作用)等函式，微調各 frame，並不斷測試，找到最適當 learning rate、threshold 的值，以達到最佳的效果。



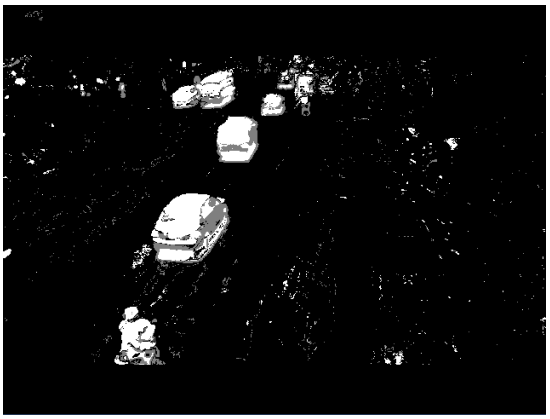
↑原始（官方版本）



↑去除雜訊後

Threshold 的選擇 ：

T=8



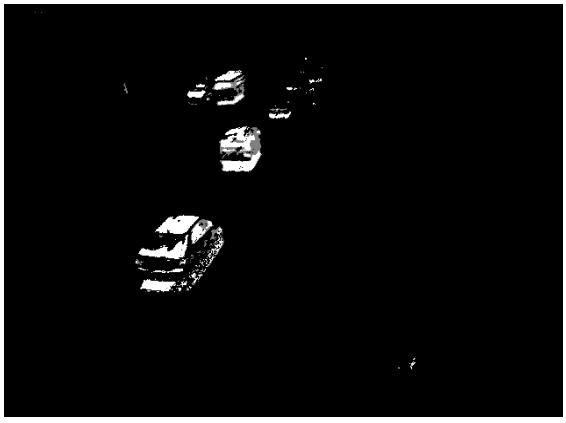
T=16

T=32

T=64

T=128

T=256



經過多次試驗和比較之後，我們覺得 Threshold 設 64 比較恰當。

Cooperation

陳冠宏：程式

報告、投影片：

Problem description, Problems encountered, Algorithm, Gaussian Mixture Model

周彥儒：程式

報告、投影片：

Problem description, Improvement (quality, speed)