車輛視覺系統 HW1

這次的作業 1 我們使用 Codebook 來完成前景偵測，方法參考論文 Real-time foreground-background segmentation using codebook model，並且自己做了一點小修改。為了更好的學習出後景與前景的模型，這裡使用了 background model 與 cache model 來存取 codeword。對於 codeword $\boldsymbol{c}_i$ 包含參數: $\mathbf{v}_i = (\overline{R}_i, \overline{G}_i, \overline{B}_i)$ 紀錄屬於該 codeword 所有資料的平均值與 $\mathbf{aux}_i = (\breve{I}_i, \hat{I}_i, f_i, \lambda_i, p_i, q_i)$，其中 $\breve{I}_i, \hat{I}_i$ 為屬於這個 codeword 資料的最小與最大亮度，為這個 codeword 出現的次數，λ_i 紀錄該 codeword 不再出現的最長時間，p_i 為 codeword 第一次出現的時間，q_i 則為最後一次出現的時間。

對於圖片中的每個 pixel，當第 t 個 RGB 資料 $\boldsymbol{x}_t = (R, G, B)$ 進來時，會先在 background model 當中尋找屬於他的 codeword $\mathbf{c}_t$，需要滿足的條件包括:顏色失真 $(\boldsymbol{x}, \boldsymbol{c}_i) \le \varepsilon$ 與亮度範圍 $I_{low} \le \|\mathbf{x}_t\| \le I_{hi}$。失真計算方法為

$$colordist(\mathbf{x}_t, \mathbf{v}_i) = \delta = \sqrt{\|\mathbf{x}_t\|^2 - p^2}, \tag{1}$$

即計算新資料點到 codeword 平均資料點方向上的垂直距離，其中

$$p^2 = \|\mathbf{x}_t\|^2 \cos^2\theta = \frac{\langle \mathbf{x}_t, \boldsymbol{v}_i \rangle^2}{\|\mathbf{v}_t\|^2}. \tag{2}$$

亮度範圍限制 $I_{low} = \alpha \hat{I}$ 與 $I_{hi} = \min\{\beta \hat{I}, \frac{\breve{I}}{\alpha}\}$，其中 $\alpha < 1, \beta > 1$。為容許新資料與先前 codeword 平均值的亮度差異。若有在 background model 找到新資料所對應的 codeword，則可以對該 codeword 參數進行更新

$$\mathbf{v}_m \leftarrow \left(\frac{f_m \overline{R}_m + R}{f_m + 1}, \frac{f_m \overline{G}_m + G}{f_m + 1}, \frac{f_m \overline{B}_m + B}{f_m + 1}\right). \tag{3}$$

$$\mathbf{aux}_m \leftarrow \left\langle \min\{I, \breve{I}_m\}, \max\{I, \hat{I}_m\}, f_m + 1, \max\{\lambda_m, t - q_m\}, p_m, t \right\rangle. \tag{4}$$

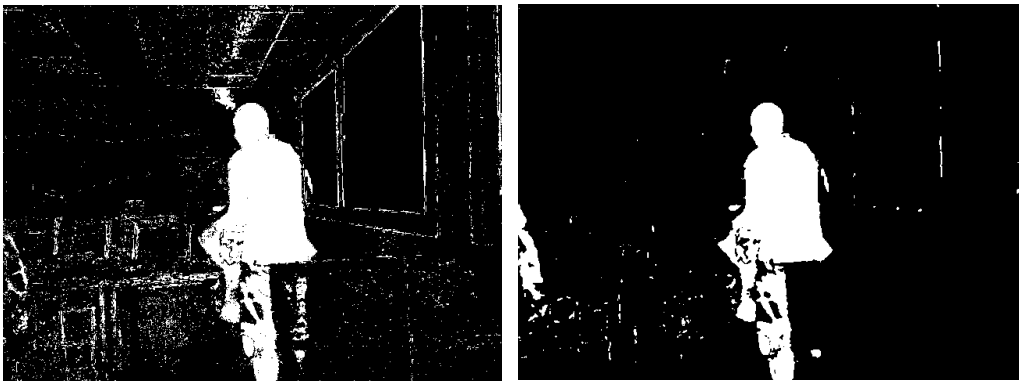
若無法在 background model 當中找到符合的 codeword，則改到 cache model 尋找，使用相同的判斷與更新方法。如果仍然無法在 cache model 中找到符合的，則在 cache model 中建立新的第 L 個 codeword，其參數為

$$\mathbf{v}_L \leftarrow (R, G, B). \tag{5}$$

$$\mathbf{aux}_m \leftarrow \langle I, I, 1, t-1, t, t \rangle. \tag{6}$$

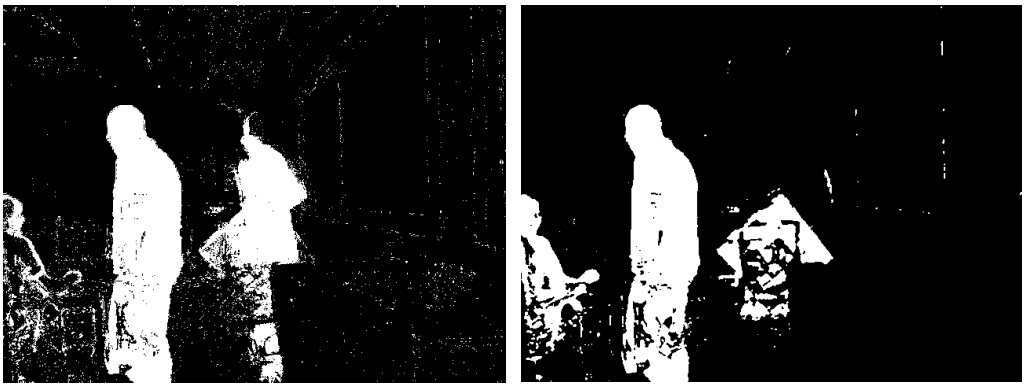
如此便將新資料加入 cache model 當中。若某個前景持續出現，即他存在 cache model 當中足夠長的時間，則可判斷為後景，便將他改放入 background model 當中。判斷方法為 $f_m \geq T_{add}$ ，即出現的次數大於設定的閥值 T_{add} 。相對的，如果在 background model 或 cache model 當中有 codeword 持續相當長的時間不再出現，則將其刪除，判斷方法為 $t - q_m \geq T_{delete}$ ，即連續未出現的次數大於設定的閥值 T_{delete} 。

透過上述方法，便可以做到後景模型的學習與建立。實作後發現了幾點問題，並且先後做了修正。首先發現做出來的圖有非常多的雜訊，也許是光線的微弱變化，於是我加入了 median filter 對圖片做修正，也就是挑選在該 pixel 周圍包含自己的 9 個點上尋找中間值的(R,G,B)值，並採用它來濾除雜訊，其結果如下圖



圖一 第 300 次輸出結果。左圖為處理前，右圖為使用 median filter 後的結果

可以明顯地看到窗戶和桌子的輪廓幾乎從圖上消失，可見 median filter 對雜訊有很好的濾除效果。然後又發現到當人走到畫面中停滯太久時，原本的存放在 background model 的背景會因為過長時間未曾出現而被刪掉，導致殘影出現，後來便將參數 T_{delete} 調整更大，便解決的這個問題。其輸出結果如下:



圖二 第 346 次輸出結果。左圖為可以明顯看到殘影，右圖則完全看不到，並且保留了人放在那裏的物品。

最後便是物品會消失的問題，原本以為這次作業希望當物品放在桌上夠長時間後，演算法要將它判斷成背景，後來看了 ground truth 的圖片才發現物品應該一直被判斷為前景，所以我又對 T_{add} 做加大，讓位於 cache model 中的前景不要太快被被判斷成後景，以解決物品會消失的問題，這也是為何前面輸出的 251 張圖片皆為空白，因為要足夠長時間才能成為後景。相對的，若今天希望物品放在桌上夠長時間後便成後景，便要將 T_{add} 改小。其輸出結果如下:

圖三 第 522 次輸出結果。左圖的物品已經消失，右圖仍可明顯看到。

最後我們再使用了 opencv 提供的函式包括: medianBlur 與 dilate，讓圖片變得更平滑且前景可以變得更連續。最後跑了助教提供的 EvaluationV2 程式，得到結果如下:

表一 EvaluationV2 結果

Recall	Specificity	FPR	FNR	PWC	Precision	FMeasure
0.87540	0.99364	0.00636	0.12460	1.86259	0.94099	0.90701