國 立 中 正 大 學

電 機 工 程 研 究 所

視 訊 處 理 作 業 一

使用光流等式實作動作估計

實作快速搜索動作估計並與 EBMA 比較

研 究 生：賴俊佑

任課教授：江瑞秋 博士

中 華 民 國 一 百 一 十 一 年 十 一 月

第一章　簡介

1.1 二維動作估計

動作估計 (Motion estimation)是影片處理系統中很重要的一部分，其中，二維動作估計除了能用於三維結構及動作估計的前處理外，還能用於影片壓縮、影片取樣率轉換和影片濾波器等等。

目前課程主要探討動作估計於影片壓縮的應用。藉由前後兩幀進行動作估測得到移動向量 (Motion Vector , MV)，利用其產生經過動作補償的預測影像，以此來最小化用於移動向量和預測誤差的總位元數。

動作估計的作法有許多種，有基於像素點進行動作估計 (Pixel-based motion estimation)、基於區塊進行動作估測 (Block-based motion estimation)、基於網格進行動作估測 (Mesh-based motion estimation)等等，每種作法都有其優缺點，而本次作業使用的是基於區塊進行的動作估測法。

1.2 區塊匹配演算法 (Block match algorithm , BMA)

在基於區塊進行的動作估測法中，最為人所知的就是 BMA，此演算法假設在同一個區塊內的所有像素點移動後，只會產生單一的 MV。因此可以分別獨立計算每個區塊的 MV，來預測整張圖片。

BMA 擁有許多種算法，如最初的全局搜索區塊匹配法 (Exhaustive block match algorithm , EBMA)，雖然可以較精確的計算 MV，但花費的計算量過大，因此衍伸出快速搜索法，如二維對數搜索法 (Two-dimensional logarithmic, 2D-log)、三步搜索法 (Three-Step Search , TSS)、新三步搜索法 (New Three-Step Search , NTSS)等等，減少計算量的同時，也有一定的精確度。

1.3 光流 (Optical flow)

在二維的影像中判斷動作的方法，主要基於時間序列影像中同一物體只是位置改變而像素點強度不變的概念。如 (1.3.1)式的位移幀差 (Displaced Frame Difference Equation , E_{DFD})等式，就是根據這個概念推導出的。

$$E_{\mathrm{DFD}}(\mathbf{a}) = \sum_{x\in\Lambda} |\psi_2\big(\boldsymbol{x} + \boldsymbol{d}(\boldsymbol{x};\boldsymbol{a})\big) - \psi_1(\boldsymbol{x})\,|^p \tag{1.3.1}$$

而光流等式 (Optical Flow Equation, E_{OF})，也是基於相同概念的假設 (1.3.2)式進行推倒，將其進行泰勒展開得到 (1.3.3)式，比較(1.3.2)式與(1.3.3)式後能得到 (1.3.4)式，並將表示式用梯度與向量替換表示成 (1.3.5)式。最後，將物體匹配問題利用上述公式轉成最小化問題 (1.3.6)式。

$$\psi\big(x + d_x, y + d_y, t + d_t\big) = \psi(x, y, t) \tag{1.3.2}$$

$$\psi\big(x + d_x, y + d_y, t + d_t\big) = \psi(x, y, t) + \frac{\partial\psi}{\partial x}d_x + \frac{\partial\psi}{\partial y}d_y + \frac{\partial\psi}{\partial t}d_t \tag{1.3.3}$$

$$\frac{\partial\psi}{\partial x}d_x + \frac{\partial\psi}{\partial y}d_y + \frac{\partial\psi}{\partial t}d_t = 0 \tag{1.3.4}$$

$$\nabla\psi^T\boldsymbol{v} + \frac{\partial\psi}{\partial t} = 0 \tag{1.3.5}$$

$$E_{OF}(a) = \sum_{x\in\Lambda} |\Big((\nabla\psi_1(\boldsymbol{x}))^{\boldsymbol{T}}\boldsymbol{d}(\boldsymbol{x};\boldsymbol{a})\Big) + \psi_2(\boldsymbol{x}) - \psi_1(\boldsymbol{x})\,|^p \to min \tag{1.3.6}$$

第二章　方法

本次作業 1 將會利用光流等式 (Optical Flow Equation, E_{OF})實作全局搜索區塊匹配法 (Exhaustive block match algorithm , EBMA)。作業 2 將會實作二維對數搜索法 (Two-dimensional logarithmic Search Method, 2D-log)、三步搜索法 (Three-Step Search Method , TSS)、新三步搜索法 (New Three-Step Search Method , NTSS)和四步搜索法 (Four-Step Search , 4SS)並與 EBMA 進行比較。

2.1 EBMA

全局搜索區塊匹配法 (Exhaustive block match algorithm , EBMA)，為最基礎的 BMA，首先，演算法會在目前幀 (Current frame) 選取一個目前要匹配的區塊 (Current block)，將 current block 對目標幀 (Target frame)搜索區域 (Search region)內的每個區塊進行搜索，計算光流等式 (Optical Flow Equation, E_{OF})，找出最小值並進行匹配，得到移動向量 (Motion Vector , MV)。如圖 2.1 範例所示。

EBMA 能找到較精準的 MV，但是會花費鉅大的計算量。

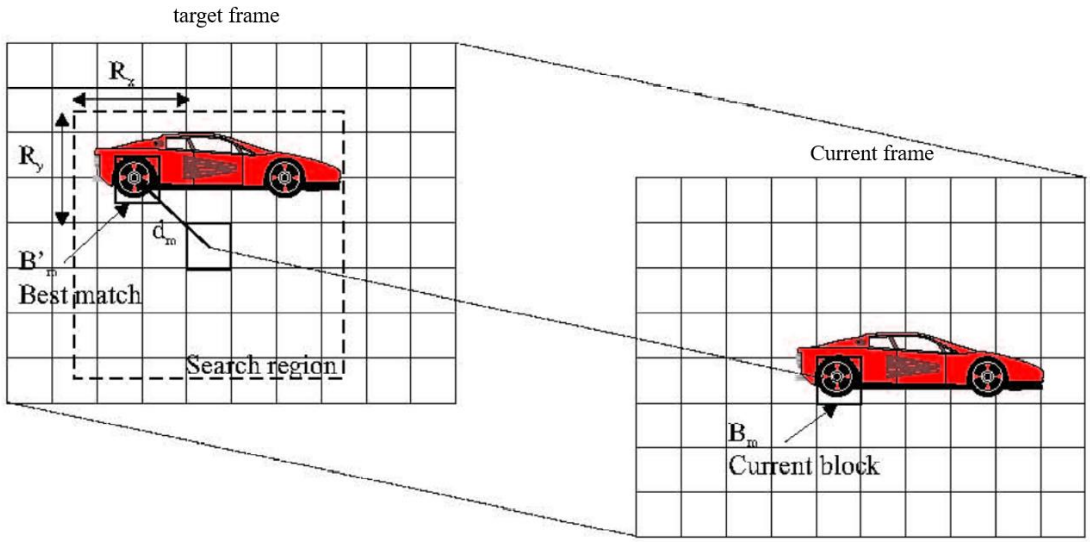

圖 2.1

2.2 2D-log

二維對數搜索法（Two-dimensional logarithmic Search Method, 2D-log)是一種快速搜索演算法，其名字源自初始搜索步長的設定(stepsize)。初始 stepsize 由(2.2.1)式和(2.2.2)式得出，其中 R 為搜索大小，n 為搜索步長。一開始會搜索中心和距離中心搜索步長的上下左右五個點，尋找最小的區塊失真測量(block distortion measure , BDM)，若是 BDM 最小值的搜索點在中心或是搜索邊界，則 step size 會減半，沒有則會繼續同樣的模式搜索，當 stepsize 等於 1 時，改為搜索中心點周圍的八個點，並選擇最小值的搜索點作為最終的匹配點。搜索範例如圖 2.2，此範例搜索範圍 R 為 6，用數字表示每一步的搜索點，用圓圈框出該步的最佳點，並用*標示最終的匹配點。

$$n' = \lfloor log_2 p \rfloor \tag{2.2.1}$$

$$n = \max\{2, 2^{n'-1}\} \tag{2.2.2}$$

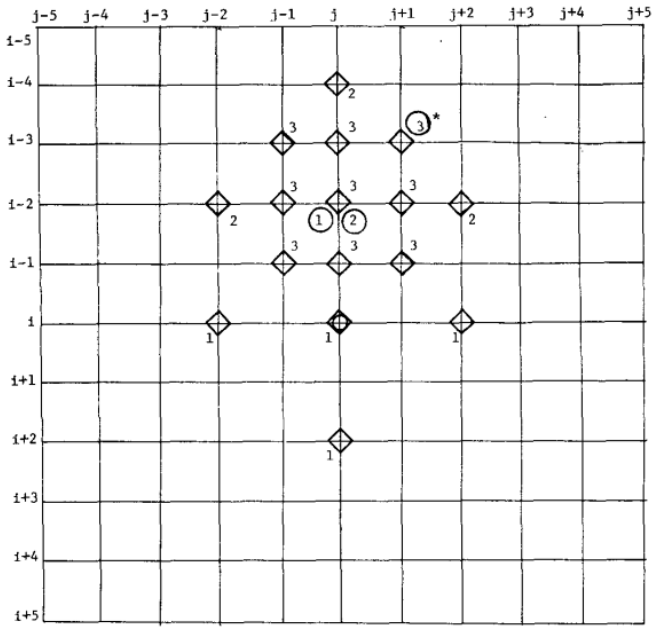

圖 2.2

2.3 TSS

三步搜索法 (Three-Step Search Method , TSS)是一種快速搜索演算法，顧名思義，此演算法在搜索範圍 R 為 6 或 7 的時候最多只需三步即可完成搜索。當搜索範圍不同時，會需要不同的步數，實際步數 L 可由 (2.3.1)式計算，(2.3.2)式可計算最大步數時需要的總搜索點數量 N。初始 stepsize 通常設為大於等於 R 的一半。

$$L = \lfloor log_2 R_0 + 1 \rfloor \tag{2.3.1}$$

$$N_{total\,search\,points} = 8L + 1 \tag{2.3.2}$$

此演算法在每一步都會搜索九個點，如圖 2.3 所示，每一步搜索完都會將 stepsize 減半，直到 stepsize 等於 1，並選取 BDM 最小值作為匹配點，計算該區塊的 MV。

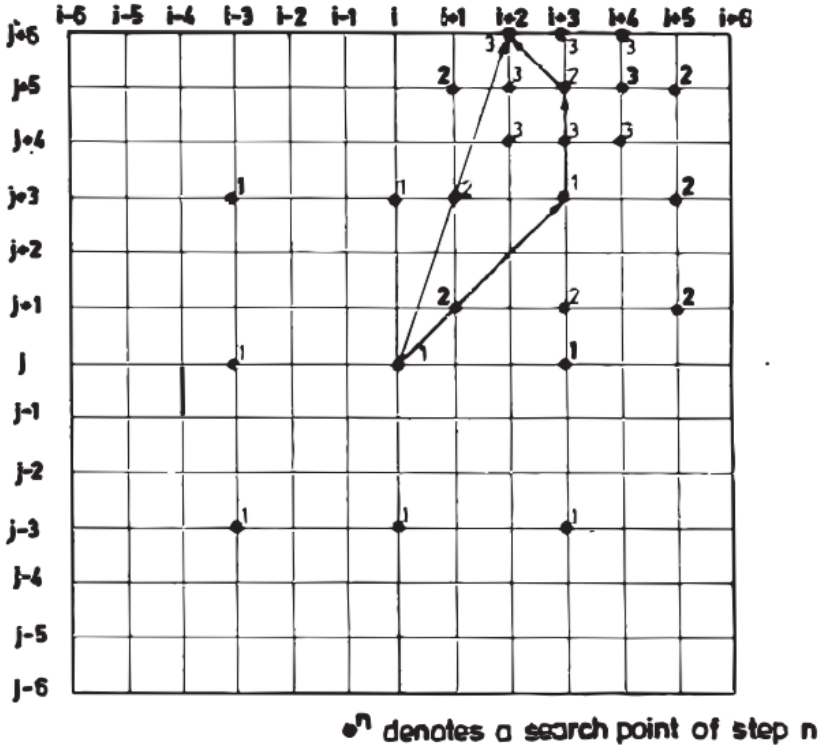

圖 2.3

2.4 NTSS

新三步搜索法 (New Three-Step Search Method , NTSS)是 TSS 的改良版，雖然 TSS 因他的簡單性和有效性十分適合用於 low bit-rate 的影像壓縮，但他在第一步使用固定分配的檢查點，導致其對於小動作的動作估計十分低效。因此 NTSS 在第一步就同時對中心區塊和外部區塊等 17 個點進行偵測，如搜索範例圖 2.4.1 中的黑色圓點與方形點，若是 BDM 最小值在中心點，則停止搜索，如果最小值在中心點鄰居的八個點內，則進行第二步，根據在角落找五個點或是邊緣找三個點，如圖 2.4.1 中的三角形點。若是最小值在最外圍的八個點內，則做 TSS 的步驟。詳細流程可參閱圖 2.4.2

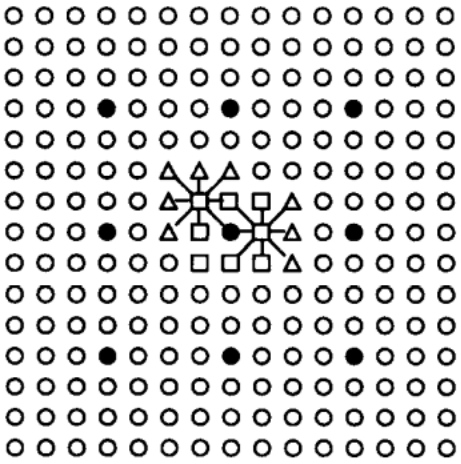

圖 2.4.1

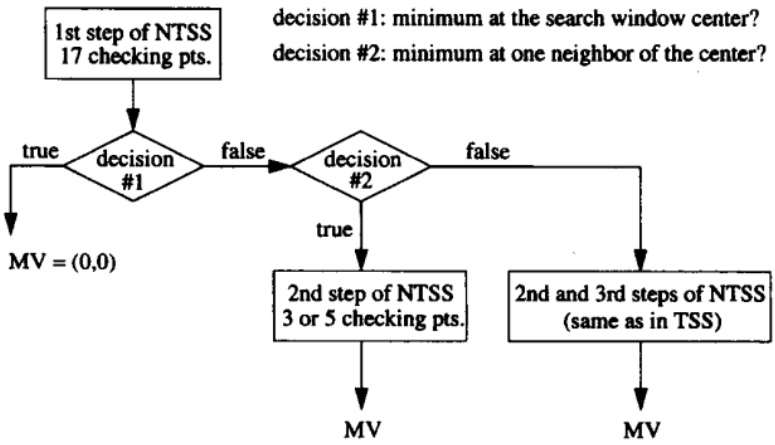

圖 2.4.2

2.5 4SS

四步搜索法 (Four-Step Search , 4SS)是基於現實世界影像序列具有中心偏移移動向量分布的特徵所提出的方法，此演算法的搜索步數為 2~4 步，總搜索檢查點為 17~27 個，與 NTSS 有相似的誤差，但 4SS 將最壞情況的檢查點從 33 個減少到 27 個，並將平均檢查點從 21 個減少到 19 個檢查點。

此演算法的搜索範例如圖 2.5，初始搜索步長設定為 2。首先找尋中心的九個點，若不是 BDM 最小值不在中心點，則根據角落或是邊緣分別搜索 3 或 5 個點，重複兩次，若是找到最小值在中心點，則將步常改為 1，尋找最小值當作翠加匹配點。

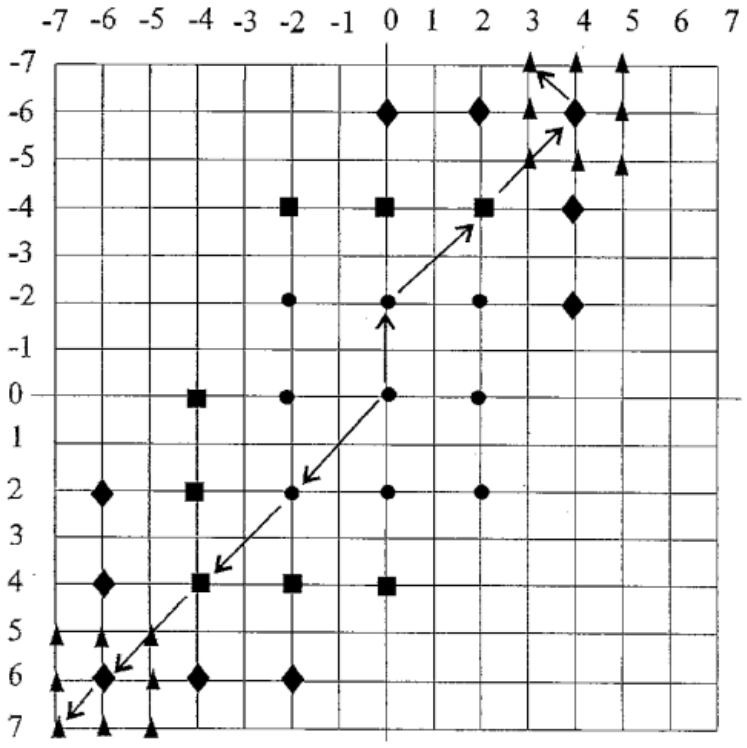

圖 2.5

第三章　實驗結果與分析

3.1 使用光流等式實作動作估計

此實驗根據作業規定，將使用(3.1)式做為區塊失真測量(block distortion measure , BDM)，並將 block size 設為 4x4。本次實驗使用第 t 幀影像當作 current frame，t+1 幀當作 target frame 進行預測。

3.2.1 比較不同大小的搜索範圍 (search range , R)

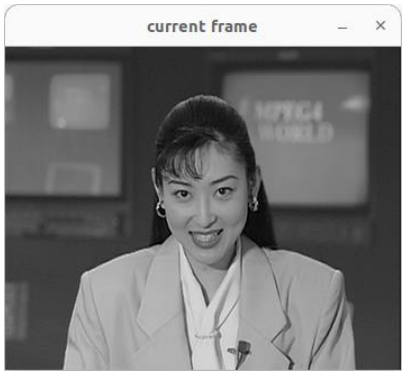

圖 3.1.1 current frame

圖 3.1.2 target frame

此處 R 設置為 8

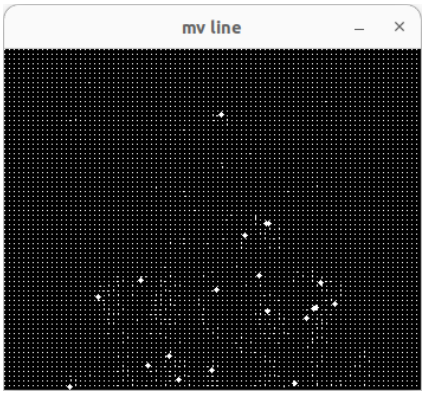

圖 3.1.3 motion field

圖 3.1.4 predicted frame

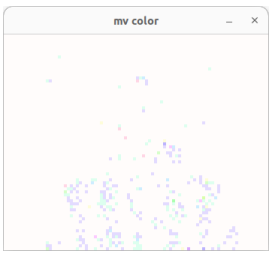

圖 3.1.5 color space　　圖 3.1.6 color motion field

此處 R 設置為 16

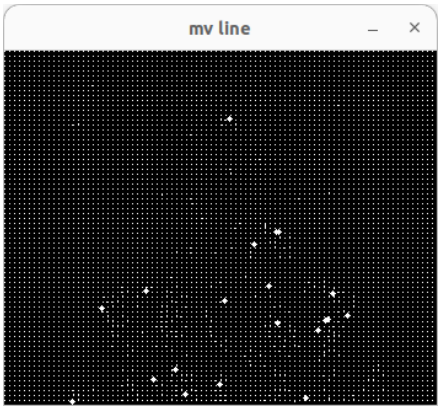

圖 3.1.5 color space　　圖 3.1.6 color motion field

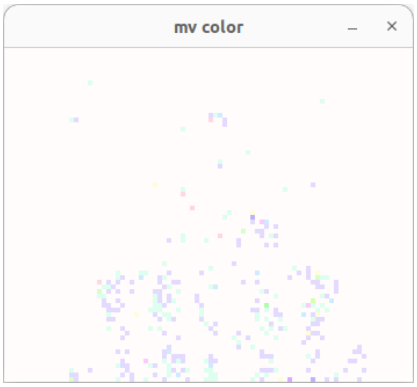

圖 3.1.5 color space　　圖 3.1.6 color motion field

此處 R 設置為 32

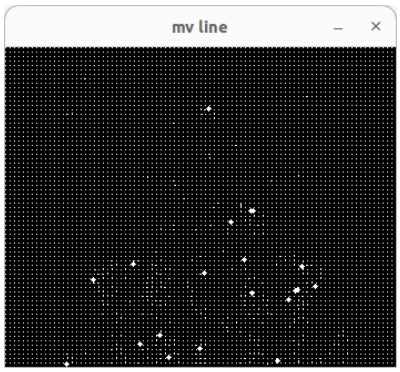

圖 3.1.3 motion field　　圖 3.1.4 predicted fram

圖 3.1.5 color space

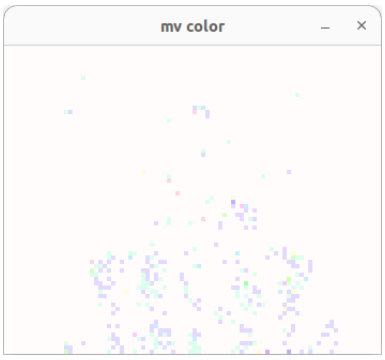

圖 3.1.6 color motion field

3.2.2 比較 E_{OF} 和 E_{DFD}

比較相同參數下，使用 E_{OF} 和 E_{DFD} 對於 predicted frame 和 target frame 間的每個像素均方誤差（Mean Square Error , MSE)有何影響。

	E_{OF}	**E_{DFD}**	**E_{OF}** Time(sec/frame)	**E_{DFD}** Time(sec/frame)
Akiyo	0.222	0.390	0.7777	0.5555
Bus	10.796	16.311	0.8888	0.6666
Container	0.416	0.672	0.8888	0.6666
stefan	5.356	9.324	0.8888	0.6666

表 3.1

由表 3.1 可看出經 **E_{OF}** 計算後的 MSE 較小，但是時間會花費比較久。

3.2 實作快速搜索動作估計並與 EBMA 比較

將(Exhaustive block match algorithm , EBMA)。二維對數搜索法（Two-dimensional logarithmic Search Method, 2D-log)、三步搜索法（Three-Step Search Method , TSS)、新三步搜索法（New Three-Step Search Method , NTSS)和四步搜索法（Four-Step Search , 4SS)進行比較。

3.2.1 比較搜索點數量及搜索步數

在 R=8 的情況

搜索演算法	搜索點的數量		搜索步數	
	最小值	最大值	最小值	最大值
EBMA	255	255	1	1
2D-log	13	32	2	7
TSS	25	25	3	3
NTSS	17	33	2	3
4SS	17	27	2	4

表 3.2.1

能從表 3.2.1 觀察到每個演算法在最糟的情況和最好的情況下分別會做多少搜索點。

3.2.2 比較各資料集的 MSE

本實驗是使用第 t 幀圖預測第 t+1 幀圖，在計算參考的圖與預測的圖的 MSE，故沒有第 1 幀的資料。且由作業一中了解 E_{OF} 較能提升精準度，故在此使用 E_{OF}。

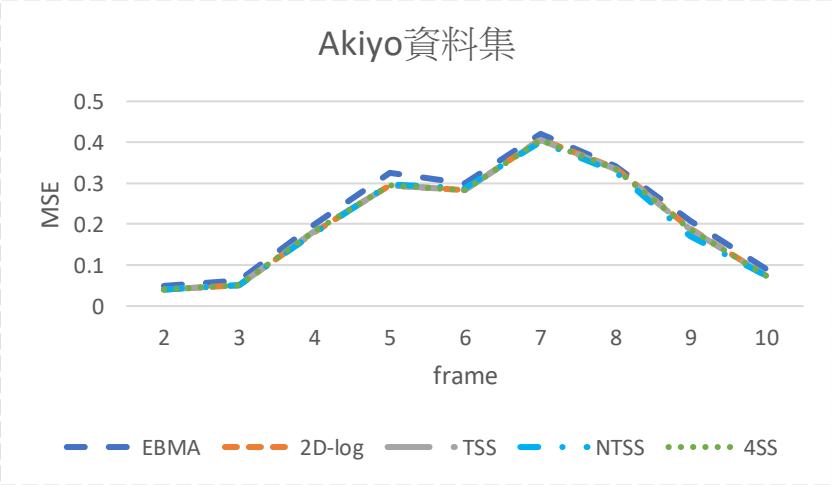

圖 3.2.1

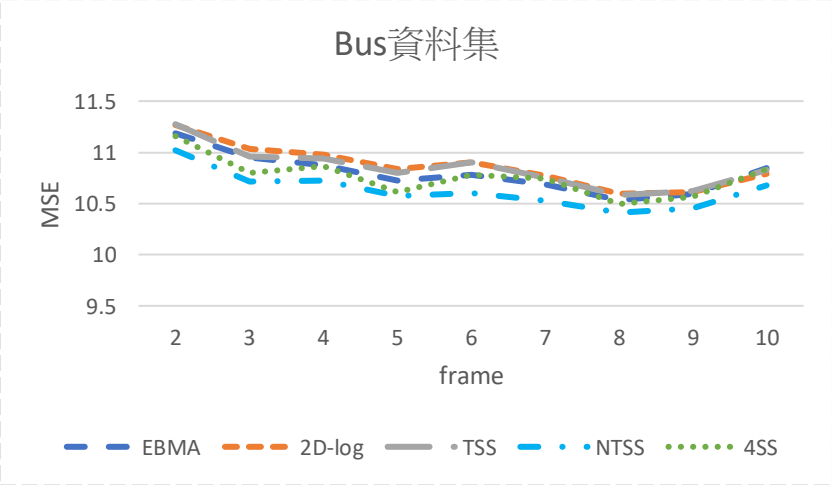

圖 3.2.2

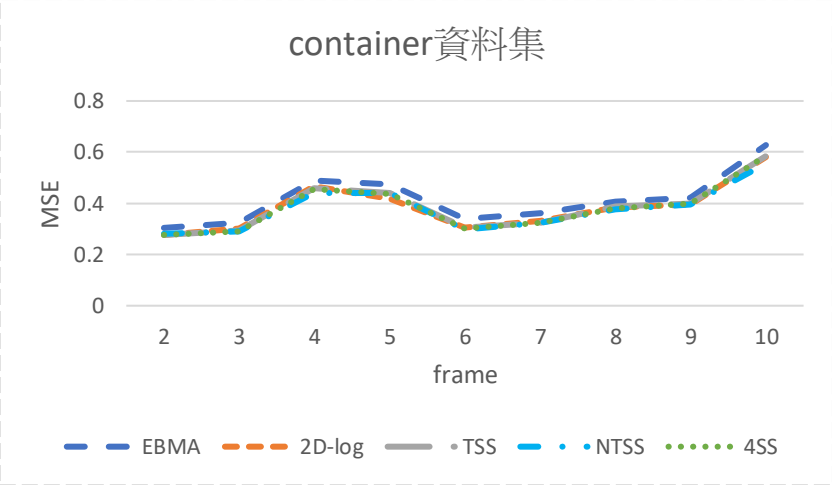

圖 3.2.3

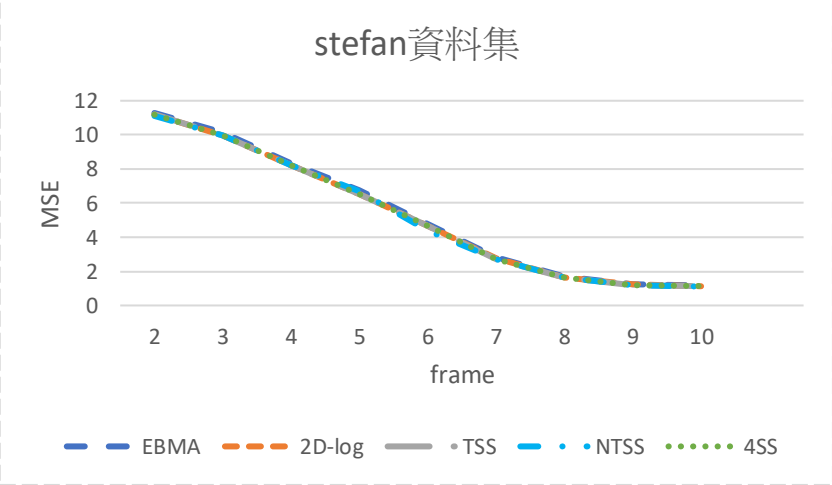

圖 3.2.4

能從圖 3.2.1 到圖 3.2.4 中觀察到，在此參數下，雖然每個演算法各有差異，但預測出的都差不多。

3.2.3 相對 EBMA，各演算法提升速度

本實驗各演算法皆使用 R=8，block size = 4x4，E_{OF} 當作衡量標準，紀錄各演算法在不同資料集的平均每幀花費的時間，以秒為單位。

	EBMA	2D-log	TSS	NTSS	4SS
Akiyo	0.7777	0.2222	0.2222	0.2222	0.2222
Bus	0.8888	0.2222	0.2222	0.2222	0.2222
Container	0.8888	0.2222	0.2222	0.2222	0.2222
stefan	0.8888	0.2222	0.2222	0.2222	0.2222

表 3.2.2

可從表 3.2.2 觀察到搜索範圍小且區塊小的情況，快速搜尋法花費的時間都差不多。

3.2.4 各演算法的預測圖

本實驗各演算法皆使用 R=8，block size = 4x4，E_{OF} 當作衡量標準，比較 Bus 資料集，用不同演算法的預測結果。

圖 3.2.5 current frame

圖 3.2.6 target frame

EBMA

圖 3.2.7 predicted frame

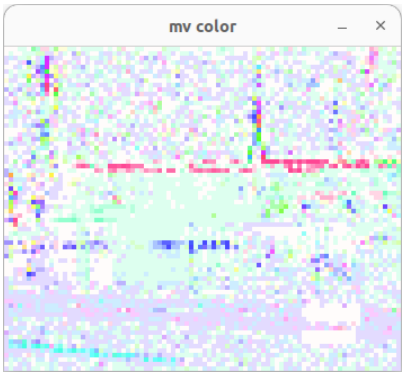

圖 3.2.8 color motion field

2D-log

圖 3.2.9 predicted frame

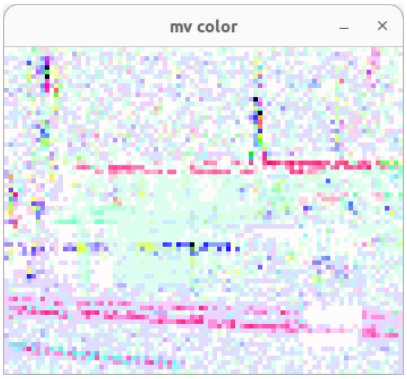

圖 3.2.10 color motion field

TSS

圖 3.2.11 predicted frame

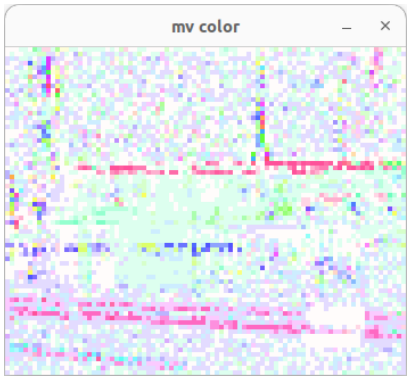

圖 3.2.12 color motion field

NTSS

圖 3.2.13 predicted frame

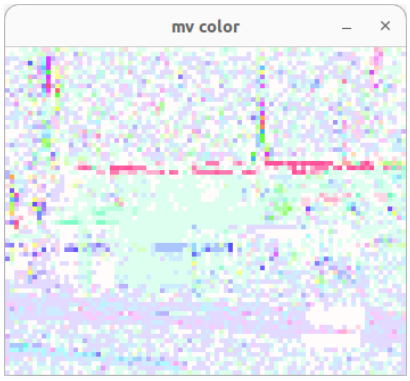

圖 3.2.14 color motion field

4SS

圖 3.2.15 predicted frame

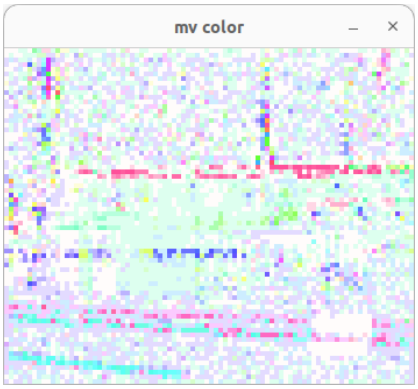

圖 3.2.16 color motion field

能從圖 3.2.7 到圖 3.2.16 觀察到雖然會有顏色的深淺，但是大致上移動向量都差不多，每種演算法的效能沒有顯著的差異。

第四章　結論

從本次作業中，學實際將這些演算法寫出來，並在查找資料的過程中更加的了解運作原理。而從我的目前的實作結果可以觀察到，E_{OF} 和 E_{DFD} 都有其優缺點，就看要用要到什麼場合。而本次比較的快速演算法都比 EBMA 還要快許多並且都有差不多的效能，我想可能是因為搜索範圍和區塊大小的原因，也許下次可以調整這部分參數再深入研究。

參考文獻

[1] Y. W. and J. O and Y. Q. Z. VIDEO PROCESSING AND COMMUNICATIONS, Prentice-Hall ,2002.

[2] J. R. Jain and A. K. Jain. Displacement measurement and its application in interframe image coding. *IEEE Trans. Commun.*, Com-29:1799-1808, Dec. 1981.

[3] T. Koga eg al. Motion-compensated interframe coding for video conferencing. In *Proc. Nat. Telecommun. Conf., pages* G5.3.1-G5.3.5, New Orleans, LA, Nov.1981.

[4] R. Li , B. Z. and Ming L. Liou, senior Member. A New Three-Step Search Algorithm for Block Motion Estimation. *IEEE Trans.* Aug 1994

[5] Lai-Man Po and Wing-Chung Ma. A Novel Four-Step Search Algorithm for Fast Block Motion Estimation. *IEEE Trans.* JUNE 1996