

國立中正大學
電機工程研究所
視訊處理作業一

實作動作估計並使用 Rate Distortion Optimization

計算分割區塊成本與 EBMA 比較

研 究 生：傅冠豪

學號：612415078

任課教授：江瑞秋 博士

中 華 民 國 一 百 一 十 二 年 十 一 月 五 日

第一章 簡介

1.1 動作估計(Motion Estimation)

動作估計 (Motion estimation)在 Video Processing System 中扮演著關鍵的角色。它不僅用於前處理已進行三維結構的動作估計以預測前後兩幀的差異，還廣泛用於影片壓縮、影片取樣率轉換等等。

在當前的課程，專注於探討動作估計於影片壓縮中的應用。利用這些向量進行動作補償生成預測影像(Predicted frame)，以此可以將移動向量和預測誤差的位元總數和最小化，提升視訊編碼效率。

我們專注於動作估計基於區塊進行動作估測 (Block-based motion estimation)，並引入了 Rate Distortion Optimization(RDO)，通過優化位元率(Bit-Rate)和失真(Distortion)之間的權衡找到最佳的編碼策略，比起只單純利用 Displacement Frame Difference(DFD)所計算的誤差，本文的方法更具有優勢，因為能更好地平衡視訊品質和壓縮效率，以提供更高圖像品質的壓縮同時保持更低的位元率。

1.2 區塊匹配演算法 (Block Matching Algorithm , BMA)

所有的動作估測方法中，最為人所知的就是基於區塊進行的 BMA，此演算法將一個 frame 完整分割了數個 block，並將區塊遍歷周圍的區塊，找出最小誤差的區塊位置，只會產生一組各自區塊獨立的 Motion Vector，用於預測整個區塊。

雖然可以透過窮舉法取得精確的移動向量 MV，但花費的計算量大，且也沒有考慮到壓縮所需要的位元數，因此本文透過將 Block 分割利用 RDO 比較兩者之間所需要的壓縮成本(cost)，將 MV 以及 Residual 的 Bits 數一併考慮，雖增加

了計算量卻提升了 Predicted Frame 的品質。

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的位置改變，但是不改變其光線的強度(luminance)，因此我們可以藉此來預測物體的移動方向。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。

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

1.4 速率失真優化演算法(RDO)

RDO 演算法的運作方式包括嘗試不同的編碼和失真選項，以找到最佳的壓縮設置，以便達到所需的壓縮率之外又不失視訊的品質，最小化失真。通過在速率和失真之間找到最佳的權衡，RDO 確保了在有限的壓縮率下獲得最佳的視頻、音頻或圖像質量。

本文會藉由 Quadtrees 分割 Block 成子區塊，利用 RDO 計算每個區塊的 cost 藉此來比較 Block 是否要被分割。

第二章 方法

本文將會利用位移幀差(Displaced Frame Difference Equation, E_{DFD})實作全局搜索區塊匹配法 (Exhaustive block match algorithm, EBMA)，除此之外會使用殘差四叉樹轉換(Residual Quadtree Transform, RQT)將區塊分割成四個子區塊，並引入 Rate Distortion Optimization 計算失真以及編碼的成本，藉由成本比較大區塊與子區塊決定該區塊是否會被分割，此算式通過調整拉格朗日乘數法(Lagrange Multiplier)平衡壓縮率以及視訊品質。

2.1 EBMA

全局搜索區塊匹配法 (Exhaustive block match algorithm, EBMA)，為最基礎的 BMA 之一，演算法會在當前幀 (Current frame) 均勻分割特定的 Block，通過迭帶依序選取一個目前要匹配的區塊 (Current block)，設定搜索區域(Search, region)(R_x, R_y)將為搜索區域內的所有集合，將 current block 對參考幀 (Reference frame)(R_x, R_y)內的每個區塊進行搜索位移幀差(E_{DFD})，找出最小失真值進行匹配，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，但存在些缺點，它不考慮可變動的區塊大小，也沒有考慮區塊的編碼成本。

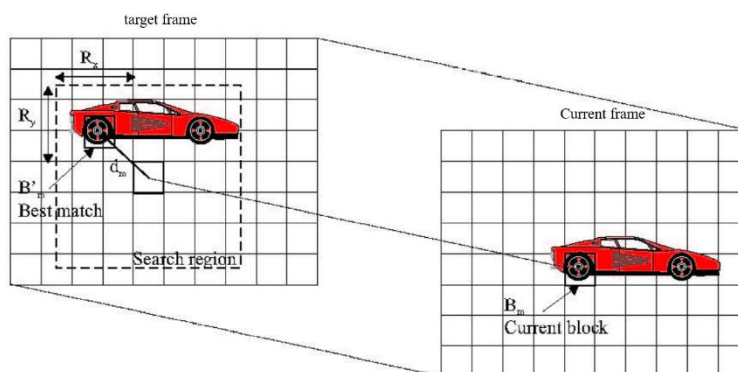


圖 2.1 EBMA 流程視覺化圖

2.2 Rate Distortion Optimization (RDO)

速率失真優化演算法(RDO)是一種優化 BMA 的方法，由於 EBMA 所分割的 Block 都是固定的，導致物體在移動時，BMA 的方法中也沒有考慮到編碼時所需要的 Bit 數量，因此本文提出了改進方案，藉由殘叉四分樹(Residual Quadtree)分割 BMA 的 Block 成 4 個 sub-Block，利用 RDO 計算 4 個 sub-Block 的 DFD 失真以及編碼所有 motion vector 以及殘叉(residual)所需要的數量，接著平均所有 sub-Block 的成本(cost)並與 Block 的成本做比較，若 Block 的成本較 sub-Block 成本高，則該 Block 就可以被分割，反之則維持原來的大小。

本文所使用的 EBMA 方法，會將輸入的 current frame 完整分割成數個 64×64 Block，搜索範圍(search range)調整為 $(\text{Block_size} // 8, \text{Block_size} // 8)$ ，位移幀差(DFD)則根據情況而定，使用絕對平均誤差(MAD)或者均方誤差(MSE)對於不同時段的 current frame 以及 reference 都有不同的效果。

RQT 具體的分割步驟可以藉由 recursive 實現，如圖 2.2 所式：

(A→C)： EBMA 迭帶到某一個 Block 時，遞迴將 64×64 block 分割成 4

個 32×32 sub_block，將 4 個 32×32 block 在各自分割 4 個 16×16 sub_block

(C→D)： 計算 4 個 sub_block 的 cost，若 sub_cost 較低就分割，反之則 Block 保留。

(D→E)： 最後計算 64×64 block 與 32×32 sub_block 的 cost，e.g.

$\frac{J_{11}+J_{12}+J_{13}+J_{14}}{4}$ 與 J_1 比較，取 cost 成本最低的 Block 作為目標。

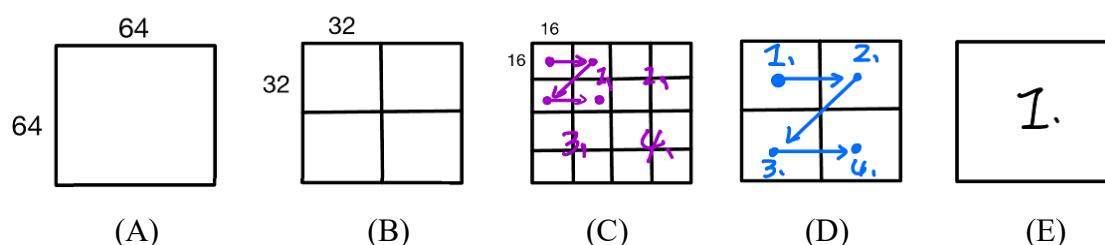


圖 2.2 RQT 分割視覺化圖

本文使用 RDO 作為計算每個 Block 的最小成本的方法，以達到權衡視訊品質以及壓縮率，算式如(2.2.1)表示：

$$J = D + \lambda R \quad (2.2.1)$$

D 定義為位移幀差(DFD)，透過 EBMA 所計算的最小誤差通過比較 current block 與對應的 reference 計算其中某個 pixel luminance 的最小誤差而得， $p = 0$ 表示為 MAD； $p = 1$ 表示為 MSE，算式如(2.2.2)表示：

$$D = \sum_{x \in \Lambda} |\psi_2(x + d(x; a)) - \psi_1(x)|^p \quad (2.2.2)$$

R 定義為位元率，通過 EBMA 計算得到 reference 最小失真的 block 的 Motion Vector，我們將 current block 通過位移到 reference block，將兩 block 中所有像素值的殘差(Residual)相加，在計算出該殘差值的 entropy，即可得知編碼此 block 所需要的位元數(Binary Bits)，算式如(2.2.3)表示：

$$R = Bit_{res} = \lceil \log_2 \sum_1^n |block_{current} - block_{ref}| \rceil \quad (2.2.3)$$

λ (Lagrange Multiplier)被定義為一個 constant value， λ 值的變動會導致視訊編碼時不同的壓縮效率和視訊品質，越高的 λ 值表示更重視失真的最小化，而越低的 λ 值則更加重視位元率的最小化。本文將 λ 值設定在 1-10 之間，如算式(2.2.4)所示，故當 λ 越傾向 10 則系統會減少子區塊的分割，儘管位元率增加但可以減少視訊的失真，反之亦然。

$$\lambda \in [1, 10] \quad (2.2.4)$$

2.3 峰值信噪比(Peak Signal-to-Noise Ratio, PSNR)

本文使用 PSNR 來衡量兩張 frame 的品質如算式(2.2.5)所示，是一種通用可以用來評估 current frame 以及 predicted frame 之間的相似程度。PSNR 的數值以分貝(dB)為單位，數值越高表示 frame 的品質越好，通常大於 40dB 可以表示是兩張相似的圖片。 $\max_intensity$ 表示 frame 的最大強度值，通常為 255；MSE 表示兩 frame 的均方誤差。

$$PSNR = 20 \cdot \log_{10} \left(\frac{\max_intensity}{\sqrt{MSE}} \right) \quad (2.2.5)$$

第三章 實驗結果與分析

本文使用 EMBA 作為 baseline，並引入了 RDO 作為 Distortion 以及 Rate 之間的權衡，從中修改 RDO 算式內部的參數如 DFD, λ 等等評估結果，最後會與 baseline 做比較。

Dataset 是 0005 以及 0033，每個 Dataset 包含了 7 個 Frame。

0005 Dataset 是一位青少年穿著溜冰鞋在一個平台上面往左滑動，camera 也隨著青少年的方向一起移動，移動幅度**偏大**；

0033 則是一位播報員的報導，只有些許的面部以及手部的運動，camera 只有及小幅度的移動。本文透過使用 color space 來表示 motion vector 可明顯的看到物體或者背景移動的方向，所使用的 Color Space 由圖 3 所示。

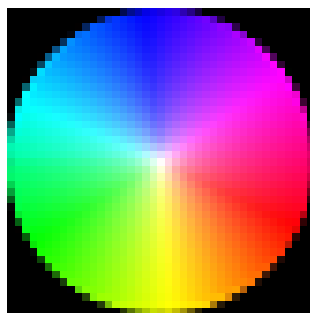


圖 3 color_space

此次的實驗內容包含以下幾點：





- 使用 MAD 進行 Motion Estimation，計算 PSNR，調整 frame 的順序
- 使用 MSE 進行 Motion Estimation，計算 PSNR，調整 frame 的順序
- 調整 Lagrange Multiplier，計算 PSNR
- 固定所有參數並比較 baseline 的結果

3.1 使用 MAD 進行 Motion Estimation

此實驗使用 Mean Absolute Deviation 作為預測 distortion 的指標，並使用 RDO 分割 Block 比較之間的成本，0005 作為實驗資料集，search range 固定為 block size/8， λ 固定為 3。

本次實驗使用第 t 幀當作 current frame， $t+1$ 幀當作 target frame 進行預測並顯示在表格中， $t+=2$ ，並且會展示從第 4 幀預測第七幀，計算預測前&預測後的 PSNR 值，分別以 $PSNR_{cur}$ ， $PSNR_{pred}$ 表示。

可以發現大部分的區塊皆有精準的描繪場景的移動，尤其是在 frame 中的物件若移動幅度較大，分割成小區塊可以提升預測的效果。

第一幀預測第二幀			
			
圖 3.1.1 current frame	圖 3.1.2 reference frame		
			
圖 3.1.3 motion vector	圖 3.1.4 predicted frame		
$PSNR_{cur}=33.4422$ dB $PSNR_{pred}= 35.1211$ dB			

第三幀預測第四幀



圖 3.1.5 current frame



圖 3.1.6 reference frame

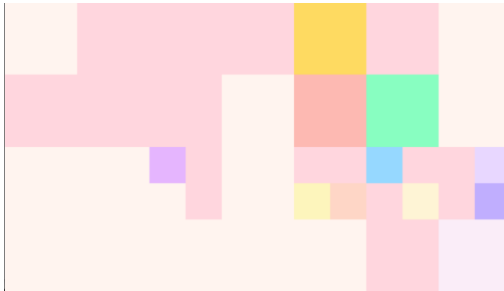


圖 3.1.7 motion vector



圖 3.1.8 predicted frame

$$PSNR_{cur}=34.8227 \text{ dB} \quad PSNR_{pred}=35.1961 \text{ dB}$$

第四幀預測第七幀



圖 3.1.9 current frame



圖 3.1.10 reference frame



圖 3.1.11 motion vector



圖 3.1.12 predicted frame



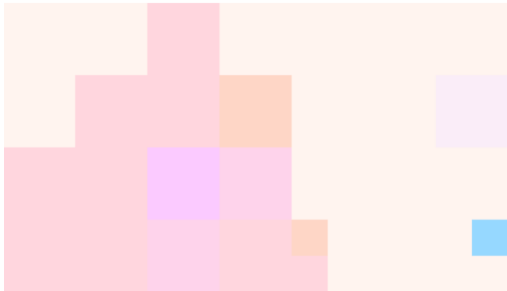

$$PSNR_{cur}=31.7406 \text{ dB} \quad PSNR_{pred}=36.0895 \text{ dB}$$

3.2 使用 MSE 進行 Motion Estimation

此實驗使用 Mean Absolute Deviation 作為預測 distortion 的指標，並使用 RDO 分割 Block 比較之間的成本，0033 作為實驗資料集，search range 固定為 block size/8， λ 固定為 3。

本次實驗使用第 t 幀當作 current frame， $t+1$ 幀當作 target frame 進行預測並顯示在表格中， $t+=2$ ，並且會展示從第 4 幀預測第七幀，計算預測前&預測後的 PSNR 值，分別以 $PSNR_{cur}$ ， $PSNR_{pred}$ 表示。

可以發現使用 MSE 比較不會出現分割小區塊的情況，若遇到較靜態的 Video Object 較沒有移動，camera 也移動幅度不大，此情況很適合使用 MSE 提升 frame 的品質。

第一幀預測第二幀			
			
圖 3.2.1 current frame	圖 3.2.2 reference frame		
			
圖 3.2.3 motion vector	圖 3.2.4 predicted frame		
$PSNR_{cur}=33.9547$ dB $PSNR_{pred}= 36.6748$ dB			

第三幀預測第四幀



圖 3.2.5 current frame



圖 3.2.6 reference frame

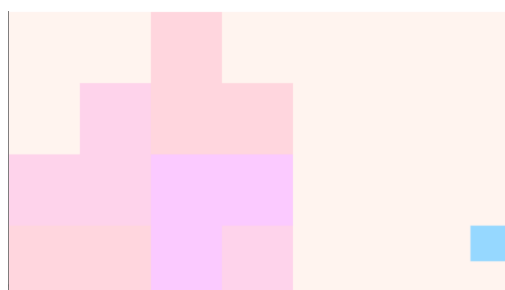


圖 3.2.7 motion vector



圖 3.2.8 predicted frame

$$PSNR_{cur}=33.8406 \text{ dB} \quad PSNR_{pred}=36.7191 \text{ dB}$$

第四幀預測第七幀



圖 3.2.9 current frame



圖 3.2.10 reference frame



圖 3.2.11 motion vector



圖 3.2.12 predicted frame

$$PSNR_{cur}=33.1431 \text{ dB} \quad PSNR_{pred}=35.2649 \text{ dB}$$

3.3 調整 Lagrange Multiplier

透過理論我們知道 Lagrange Multiplier 越大代表更位元率會減少，但是可能導致 predicted 失真的較嚴重，我們將 0005 以及 0033 資料集的第三幀以及第四幀作為實驗數據，調整 $\lambda=1$ 以及 $\lambda=10$ ，可以發現 $\lambda=1$ 時 Block 區塊被分割的很多子區塊，但是失真較嚴重，而反觀 $\lambda=10$ ，Block 區塊被分割的少很多，失真減少但是導致位元率的增加，實驗所使用的參數與 3.1 一致。

0005_第三幀預測第四幀



圖 3.3.1 current frame



圖 3.3.2 reference frame

$$PSNR_{cur} = 35.1961 \text{ dB}$$

Lagrange Multiplier $\lambda = 1$



圖 3.3.3 motion vector



圖 3.3.4 predicted frame

$$PSNR_{pred} = 34.6733 \text{ dB}$$

Lagrange Multiplier $\lambda = 10$



圖 3.3.3 motion vector

圖 3.3.4 predicted frame

$$PSNR_{pred} = \underline{34.8568} \text{ dB}$$

0033_第三幀預測第四幀



圖 3.3.5 current frame



圖 3.3.6 reference frame

$$PSNR_{cur} = 33.8405 \text{ dB}$$

Lagrange Multiplier $\lambda = 1$



圖 3.3.7 motion vector



圖 3.3.8 predicted frame

$$PSNR_{pred} = \underline{34.6733} \text{ dB}$$

Lagrange Multiplier $\lambda = 10$



圖 3.3.9 motion vector







圖 3.3.10 predicted frame

$$PSNR_{pred} = \textcolor{red}{34.9512} \text{ dB}$$

3.4 實作快速搜索動作估計並與 EBMA 比較

由於 EBMA(baseline)是不可變動區塊的窮舉演算法，因此在遇到較複雜的 object motion 時，就會產生 block artifacts，RDO 則是可變動區塊且也可以根據需求調整 Langrange Multiplier。為了展示本文所提出的方法較 baseline 佳，本實驗透過 0005 資料集的第一幀以及第四幀作為實驗數據，比較 predicted frame 以及 PSNR 的差異。

我們發現對於有移動 camera 或者物體較大的移動時，RDO 會分割較多子區塊可以提升壓縮的效率，而 EBMA 因為區塊是固定的因此僅只能找到最匹配的區域無法達到更有效的匹配。

0005_第一幀預測第四幀	
	
圖 3.3.1 current frame	圖 3.3.2 reference frame
$PSNR_{cur} = 31.7938 \text{ dB}$	
EMBA + RDO	
	
圖 3.3.3 motion vector	圖 3.3.4 predicted frame
$PSNR_{pred} = 33.1932 \text{ dB}$	

EMBA



圖 3.3.3 motion vector



圖 3.3.4 predicted frame

$$PSNR_{pred} = \underline{32.9613} \text{ dB}$$

第四章 結論

從本次作業中，我們學會自己查找 RDO 的公式，並善用 AI, Github 等工具將公式以及程式完成，以及學會自己撰寫 recursive 將子區塊找出來並計算 cost，與大區塊進行比較。而從我的目前的實作結果，Lagrange Multiplier 的調整最讓我感到印象深刻，可以透過調整權衡 frame 的效果，透過調大讓大區塊變多，提高了效率，透過調小犧牲了一些 frame 的品質但提升編碼的效率。下次可以調整這部分參數再深入研究。

參考文獻

- [1] OpenAI. (2023). ChatGPT (Mar 14 version) [Large language model].
- [2] Yang, Y., & Hemami, S. S. (2000). Generalized rate-distortion optimization for motion-compensated video coders. IEEE Transactions on Circuits and Systems for Video Technology, 10(6), 942-955.
- [3] Github: <https://github.com/001honi/video-processing/tree/main/homework-1>