

**111 學年度第 1 學期**  
**Video Processing**

**Homework\_2**  
**Implementation of a**  
**video coding system**

姓名： 柯佐勲  
學號： 611415055  
系所： 電機所  
教師： 江瑞秋老師  
日期： 2022.12.13

## 一、簡介

這次的作業是要透過撰寫程式完成影像編碼系統(Video Coding System)，以實現對特定的影像編碼格式進行壓縮。所謂的影像編碼是影像壓縮的一種方式，將原始之輸入影像經過編碼(Encoding)處理後，在維持與輸入影像相同的品質、解析度情況下，進行壓縮成影像編碼格式，以提升影像之儲存或傳輸之使用效率。而在日常生活中我們在觀看的影片，其播放原理是在短時間內快速切換影像，透過肉眼的視覺佔留，產生連續的播放效果。然而，要在每秒鐘快速地切換多個影像，將會使原始影片檔案的數據量變得相當龐大，不僅僅是佔據了儲存空間，也不利於傳輸與播放。因此可藉由影像編碼的方式將這些原始影片資料做轉換與壓縮，使其更易於儲存、傳輸以及播放的格式，以達到提升影片播放品質與傳輸效率。

## 二、方法

本次作業之實驗方法主要流程分為，影像動作估測補償、轉換編碼、標準量化、Entropy 編碼以及最後的重建，組合成一影像編碼系統，做影像之壓縮、編碼與解碼之功能，如 Fig. 1 所示。

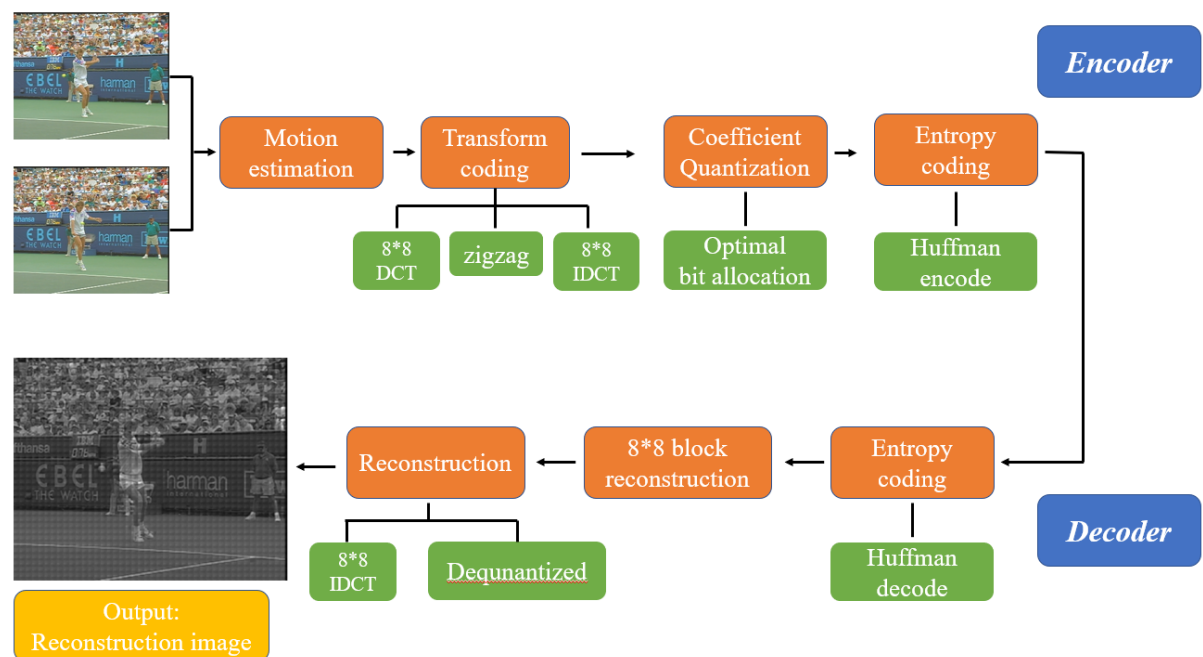


Fig. 1 VCS(Video Coding System)流程圖

第一步利用 Motion estimation and compensation 對影像中的每一幀做預測其影像中的對象/物件的像素運動方向，我使用 Exhaustive Block Matching Algorithm 對整張影像進行搜索，如 Fig.2 所示，按 block 的順序在參考影像中，進行搜索整個全域範圍，再以全域範圍內 block 以所有位置為中心與目標 frame 的 block 進行比較，最後產生 error image 做後續的影像重建方法。

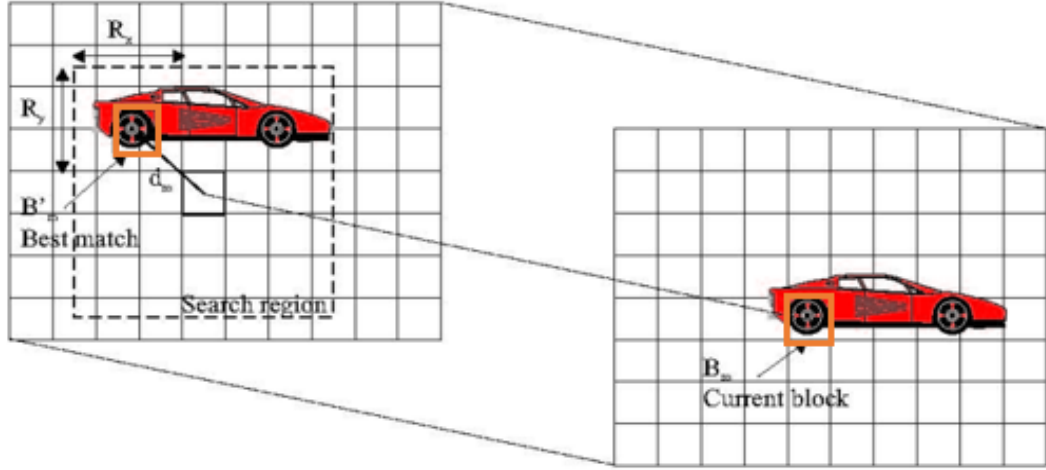


Fig. 2 EBMA 示意圖

第二步藉由運動估測所輸出的 error image，做 8\*8 之 DCT(Discrete Cosine Transform)離散餘弦轉換，經常被應用於信號或影像之處理，其功能為將影像資料先切割成 8X8 各區域，如 Fig.3 所示，影像為 8\*8 之大小，在座標(m,n)的灰階值為 $f(m,n)$ ，如公式(1-1)所示，再依據每個影像區域中的資料較不重要的部分做濾除，僅保留重要的影像資訊，以達到高效率的有損壓縮之目的。

$$2D \text{ DCT} \quad F(u,v) = \frac{1}{4} C(u)C(v) \sum_{m=0}^7 \sum_{n=0}^7 f(m,n) \cos\left(\frac{(2m+1)u\pi}{16}\right) \cos\left(\frac{(2n+1)v\pi}{16}\right) \quad (1-1)$$

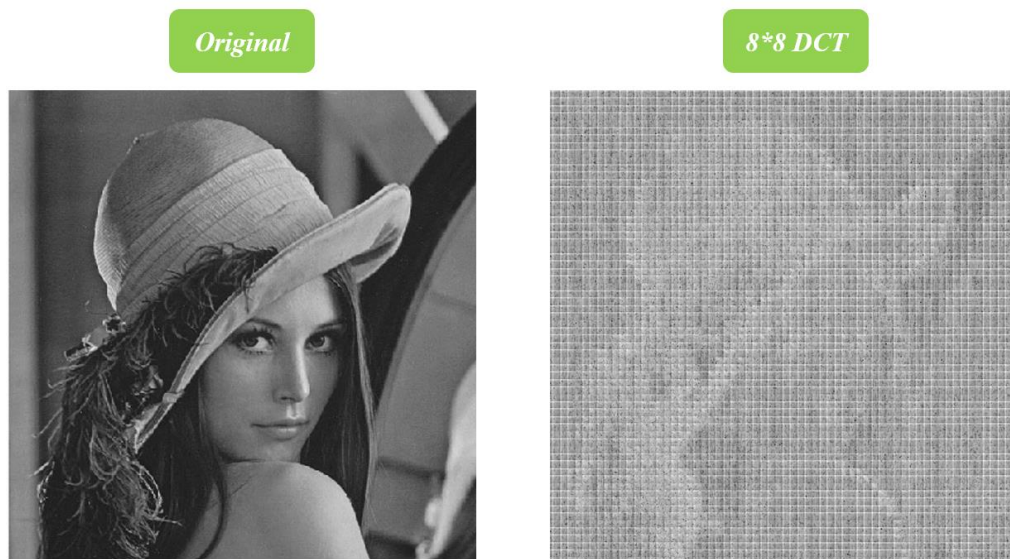


Fig. 3 實際以 Lena 影像做 8\*8DCT 之結果圖

接著透過影像壓縮技術中常使用之曲折掃描 Zig-zag scanning，如 Fig. 4 所示，以曲折方式掃描影像中所有區塊之像素值，而在本次作業中僅保留前 4 個區塊的像素，其他區塊則歸零做有限係數之反轉換，再來進行 8\*8 之 2D IDCT(Inverse Discrete Cosine Transform)反離散餘弦轉換，以及計算重建之後的 error image 之峰值訊噪比 PSNR(Peak Signal-to-Noise Ratio)，以 PSNR 之值來了解重建後之 error image 與第一步驟所產生的 error image 之間的失真量。

$${}_{2D\ IDCT}\quad f(m,n)=\frac{1}{4}\sum_{u=0}^7\sum_{v=0}^7C(u)C(v)F(u,v)\cos(\frac{(2m+1)u\pi}{16})\cos(\frac{(2n+1)v\pi}{16})\quad (1-2)$$

Perceptual based quantization matrix:

16	11	10	16	24	40	51	61
12	12	14	19	26	58	60	55
14	13	16	24	40	57	69	56
14	17	22	29	51	87	80	62
18	22	37	56	68	109	103	77
24	35	55	64	81	104	113	92
49	64	78	87	103	121	120	101
72	92	95	98	112	100	103	99

Zig-zag ordering of DCT coefficients:

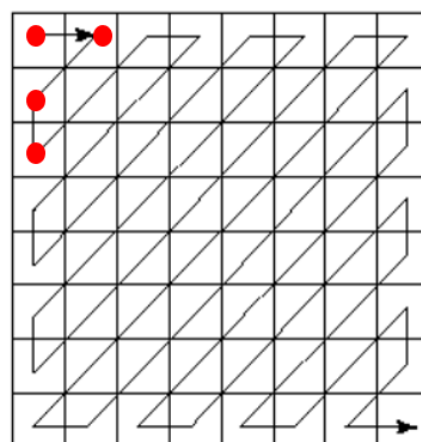


Fig. 4 Zig-zag scanning 示意圖

第三步使用在第二步中前 4 個區塊的像素值係數，以標準量化的方式做量化(Quantization)，在影像品質可接受的情況下，將不重要的訊息捨棄掉，分配其參數位元為[4,2,2,2]，做優化配置，如公式(1-3)，再以反量化(Inverse Quantization)做反轉換，來完成影像之重建。而所謂的量化主要是用來對影像進行壓縮，將影像之像素值從浮點數轉成整數，此方法為有損壓縮。

$$Q(f) = \left\lfloor \frac{f - f_{\min}}{q} \right\rfloor * q + \frac{q}{2} + f_{\min} \quad (1-3)$$

第四步將執行 Entropy coding 之運算與處理，使用最原始(步驟一)之 error image 作為輸入，並提取 error image 中的所有像素之顯示機率，以範圍-255 至 255 之直方圖做估計與顯示，並計算該 error image 的 entropy，可藉由 entropy 驗證在對影像進行壓縮時，是否完整保留在壓縮前的資訊，此目的是希望達到理想的影像資料壓縮比；Entropy coding 之第二小步透過 codeword length 建構霍夫曼編碼(Huffman code)，並依據出現機率進行大小排序，排序方式為從 496 個小符號中，當機率小於前 15 個符號時，將此 15 個符號組合成一個新符號(ESCAPE 符號)，因此以此 15 個擁有高機率之符號和 ESCAPE 符號構建一霍夫曼表；在藉由附加一個 9 位元固定長的碼在 ESCAPE 符號後，用於區分機率較低之符號與 ESCAPE 符號的 VLC(Variable length coding)代碼字表，以此達到減少 bit 的目的，最後再計算壓縮數據前與後之壓縮比率，作為評斷解碼壓縮好壞之指標。

第五步將經過去量化和反 DCT 轉換之 error images 進行運動補償，以獲得一新的重建影像，並計算其影像之峰值訊噪比(PSNR, Peak-Signal to Noice-Ratio)，如公式(1-5)所示，以驗證重建影像之精度。

$$MSE = \sigma_e^2 = \frac{1}{FrameSize} \sum_{n=1}^{FrameSize} (I_1 - I_2)^2 \quad (1-4)$$

$$PSNR = 10 \times \log_{10}(\frac{255^2}{\sigma_e^2}) \quad (1-5)$$

### 三、結果

在本次作業實驗中，我以網球選手(stefan)的影像資料集作為實驗對象，分別取第一幀與第五幀影像作為步驟一 Exhaustive search 的輸入，做運度估測補償並生成 error image，做後續的影像重建實驗。本章節將分別呈現每一步驟所得到的重建影像以及相關數據結果。

Table. 1 硬體設備與實驗環境配置





作業系統	Windows 10 64 bits
CPU	Intel® Core™ i5-8265 @ 1.60 GHz
GPU	NVIDIA GeForce MX250 2GB
IDE	MATLAB R2022b

#### 3.1 Step 1: Motion estimation and compensation

Table. 2 上方為 target frame 與 anchor frame，利用 EBMA 搜索法設置 Block size 為 16，search range 為 8，得到 predicted frame 後藉由與 anchor frame 之間的差異產生了 error image，可發現選手的臉部與腿部皆有明顯得失真情形，表示動作變化幅度較大。



Table. 2 EBMA 執行整數像素運動估計並獲得 error image

Target frame	Anchor frame
	
Predicted frame	Error image
	

### 3.2 Step 2: Transform Coding

Error image 透過公式(1-1)、(1-2)執行二維的  $8 \times 8$  DCT (Discrete cosine Transform)轉換，做影像之有損壓縮與編碼。接續做反 DCT 以及藉由 zig-zag scan 對影像中每個區塊的像素值進行掃描與取得，並透過遮罩的方式，實現 zigzag scan，只保留前 4 區塊的像素值，其餘皆改為 0。如 Fig.5 所示，可發現經過轉換與掃描後的影像係數中，在每一個  $8 \times 8$  區域之左上角區塊的係數值，均具有較大的能量，也可透過 Fig.6 做驗證與顯示，可明顯地在左上角看到縮小化的 error image。最後在實驗中計算壓縮前後影像之 PSNR，作為影像重建好壞的評估標準，實驗中得到之 PSNR 值為 19.207 db，可藉由影像觀察以及 PSNR 的值得知，在第二步驟所重建的影像是具有較多失真的區域，如 Table. 3 所示。

288x352 double								
	1	2	3	4	5	6	7	8
1	442.2500	128.9533	0	0	0	0	0	0
2	338.3265	0	0	0	0	0	0	0
3	45.7750	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0
5	0	0	0	0	0	0	0	0
6	0	0	0	0	0	0	0	0
7	0	0	0	0	0	0	0	0
8	0	0	0	0	0	0	0	0
9	77.6250	28.3296	0	0	0	0	0	0
10	-6.0471	0	0	0	0	0	0	0
11	2.8817	0	0	0	0	0	0	0
12	0	0	0	0	0	0	0	0
13	0	0	0	0	0	0	0	0
14	0	0	0	0	0	0	0	0
15	0	0	0	0	0	0	0	0
16	0	0	0	0	0	0	0	0

Fig. 5 影像經二維 8\*8 DCT 轉換以及 zigzag 後之陣列係數

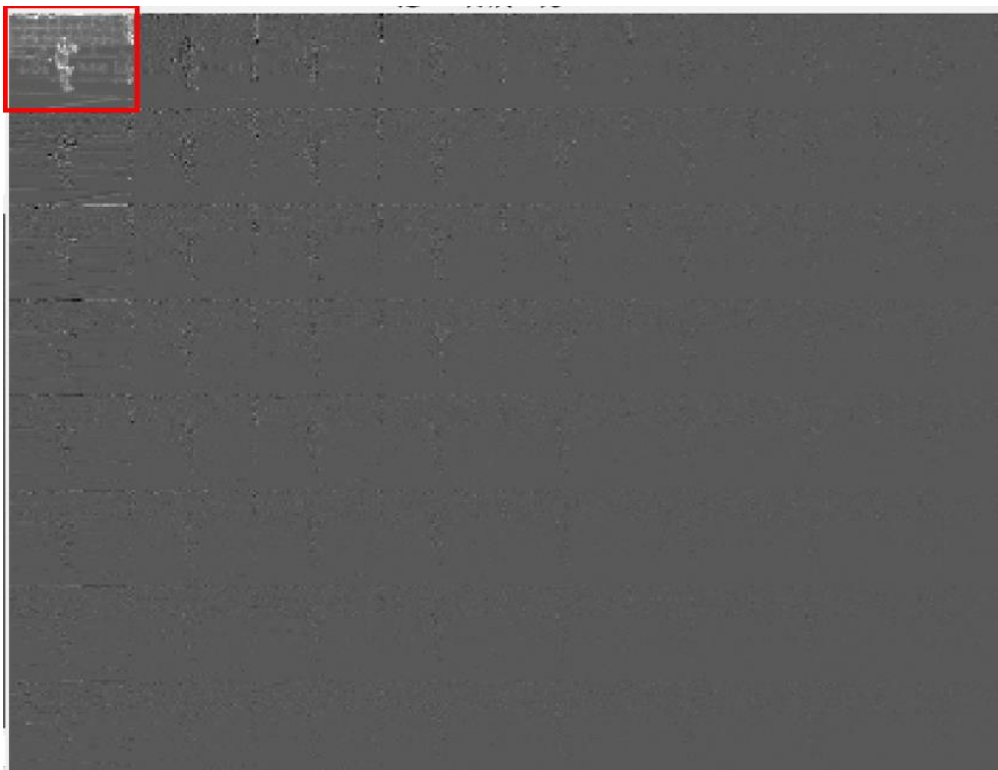
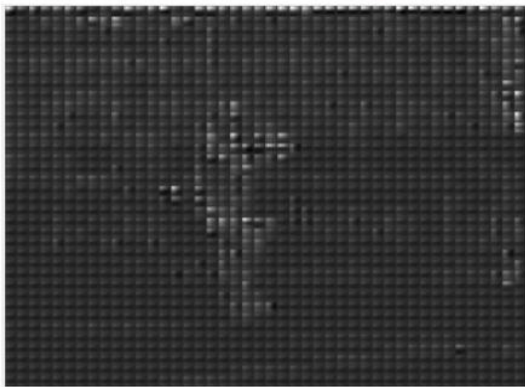



Fig. 6 影像經二維 8\*8DCT 轉換後之影像





Table. 5 做量化與反量化後之重建影像，  
以及計算其與第一步之 error image 之 PSNR 值

Reconstructed error image (Quantization 、Inverse quantization)	Error image
	
PSNR: 18.81 db	

### 3.4 Step 4: Entropy Coding

透過原始(步驟一)之 error image 作為輸入，並提取 error image 中的所有像素之顯示機率，以範圍-255 至 255 之直方圖做估計與顯示，並在 Entropy coding 實驗中計算該 error image 的 entropy 值為 0.989 bit/pixel，如 Table.6 所示；接著使用 error image 的 codeword length 建構霍夫曼編碼(Huffman code)，根據顯示機率進行大小排序，將 496 個小符號中，前 15 個擁有高機率之符號構建一 Huffman table，如 Table.7 所示，而剩餘的符號會用於組合成一個 ESCAPE 新符號，如 Table.8 所示，此用意是為了區分低於前 15 個高機率之剩餘符號。最後再計算壓縮數據前與後之壓縮比率，作為評斷解碼壓縮好壞之指標，得到之壓縮比率為 **134.5401%**。

Table. 6 Error image 之像素顯示機率直方圖與 Entropy 之值

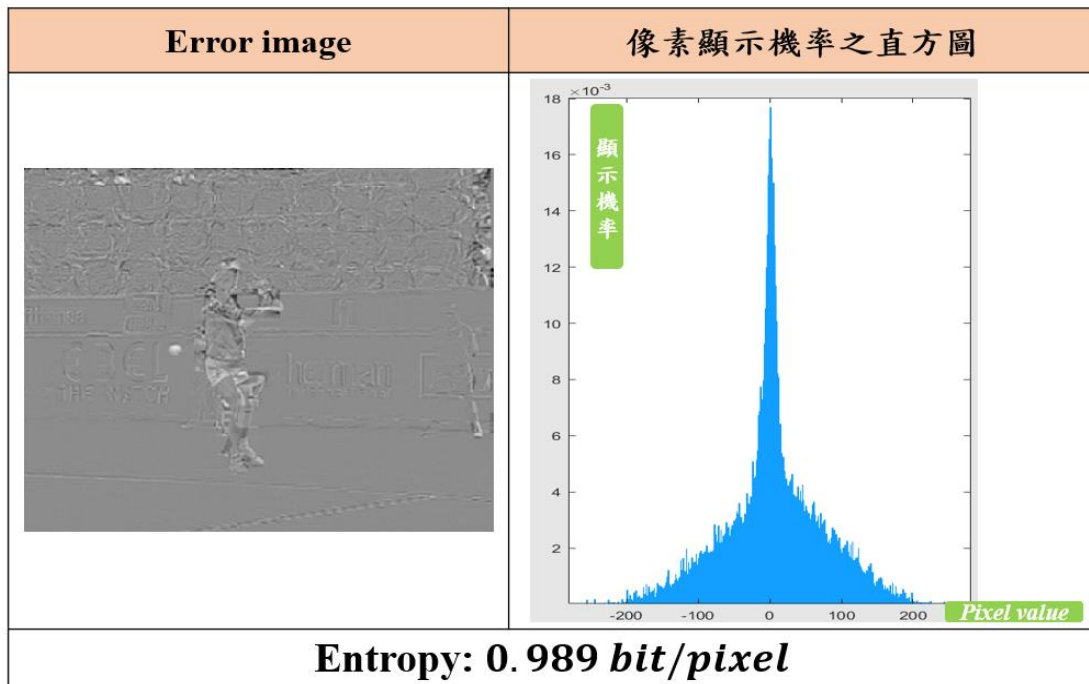


Table. 7 為前 15 個高機率符號設計一 Huffman table

	<i>Pixel value</i>	顯示機率	發生次數	<i>Number of Bit</i>
	1	2	3	4
1	0	0.1212	12284	1
2	1	0.0983	9964	3
3	-1	0.0971	9840	3
4	2	0.0648	6569	4
5	-2	0.0641	6499	4
6	3	0.0377	3.8220e+...	5
7	-3	0.0371	3763	5
8	4	0.0242	2454	5
9	-4	0.0238	2417	6
10	-5	0.0196	1.9850e+...	6
11	5	0.0184	1869	6
12	6	0.0159	1.6150e+...	7
13	-6	0.0157	1588	7
14	7	0.0148	1496	7
15	-7	0.0145	1473	7

Table. 8 為剩餘的符號(16~511)組合成為 ESCAPE 符號

	<i>Pixel value</i>	顯示機率	發生次數	<i>Number of Bit</i>		<i>ESCAPE symbol</i>
16	-8	0.0128	1298	12	Append →	1 0000010000000001
17	9	0.0120	1218	12		2 0000010000000010
18	8	0.0120	1216	12		3 0000010000000011
19	-9	0.0117	1189	12		4 0000010000000100
20	-10	0.0107	1086	12		5 0000010000000101
⋮						⋮
507	251	0	0	12	Append →	492 000001111101100
508	252	0	0	12		493 000001111101101
509	253	0	0	12		494 000001111101110
510	254	0	0	12		495 000001111101111
511	255	0	0	12		496 000001111110000

### 3.5 Step 5: Reconstruction

最後將步驟 3 未經量化之 error images 與步驟一之 target frame 進行運動補償，以獲得新的重建影像，並計算其影像之峰值訊噪比(PSNR , Peak-Signal to Noice-Ratio)，以驗證重建影像之精度，如 Fig.7 所示。

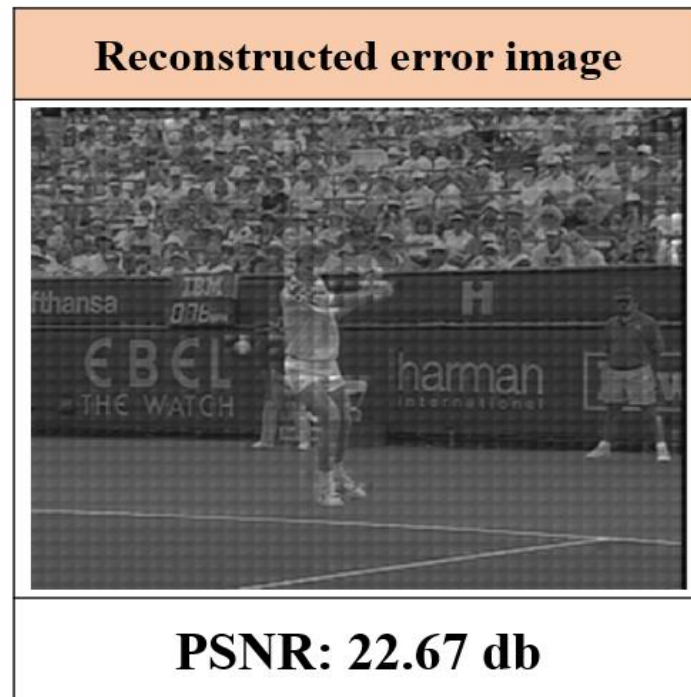


Fig. 7 重建影像

## 四、結論

這次的作業整合了本學期在 Video Processing 課堂中所學到的理論知識，關於影像壓縮、轉換、編碼等，也透過作業的方式實際以程式做出影像編碼系統，使我更加了解影像壓縮的過程，並在最後的實驗結果可得知在第 2 步之重建影像與 error image 之 PSNR 值(19.2)，以及最後重建影像之 PSNR 值(22.67)，有明顯的上升，表示失真有越來越少之趨向，更在設計 2 維 8\*8 DCT 轉換時，實際以函式以及用 zigzag 掃描的方式更改影像的像素值，有別於以往在課堂中只有用聽的，現在以程式的方式實際完成了這項功能，充滿成就感。雖然這次作業相較於作業一來說困難許多，但是在經過與同學討論後，總算是一步一步地完成了。

而影像編碼系統(VCS)的使用情境以及應用，並非單純只在影像壓縮之任務中，也可應用在機場之行李流水線中[1]，其功能為進一步解決行李箱上未成功被讀取之標籤，僅須透過 VCS 來正確地讀取與拍攝行李箱上之標籤，傳至操作員端做標籤之觀察，與排除無法準確辨識之問題，減少了機場在高峰時段期間可能出現的意外，同時提升運作效率。

## 參考文獻

[1] Take baggage tag encoding to another level with video coding :

<https://knowledge.beumergroup.com/airports/baggage-tag-video-coding-system>