國立交通大學

資訊科學系

碩 士 論 文

利用階層分解之視訊壓縮

Video Compression via Layer Decomposition

研 究 生：陳育信

指導教授：莊仁輝　教授

中 華 民 國 九 十 四 年 六 月

利用階層分解之視訊壓縮

研究生：陳育信　　指導教授：莊仁輝 博士

國立交通大學

資訊科學研究所

摘要

本論文針對現行的視訊壓縮標準 H.264，探討前景和背景分析的應用，目的在透過前景的影像品質、背景的資料量之降低，以及壓縮速度之提升等三方面，來改善整體標準的壓縮效率。本論文對於 H.264 標準中可變區塊大小和失真模糊率的兩種編碼工具進行調整與分析，並提出利用前景與背景遮罩來增加編碼效能，而這個方法亦符合 H.264 標準內所規範的語法和語意，不會有相容性的問題。實驗的結果也顯示，我們所提出的方法能相當有效的改善 H.264 的編碼效率。另外，本論文亦提出一套自動視訊切割方法可供使用者使用，可以在攝影機運動的情況下，切割出真實世界中運動的物體，實驗結果顯示，我們所提出的方法切割效果良好。

Video Compression via Layer Decomposition

Student：Yu-Shin Chen Advisor:Dr. Jen-Hui Chaung

Institute of Computer and Information Science

Nation Chiao Tung University

ABSTRACT

In this thesis we develop a method to improve the encoding performance of the H.264 video standard when the foreground and background models are given. The encoding performance is measured in three aspects: the PSNR, bit-rates and coding speed. By analyzing the variable block-size and rate-distortion coding tools of H.264, the proposed methods can enhance the video quality of the foreground objects and reduce the compression bit-rates of the background area. The syntax and semantics of the encoded bit-streams generated by our method also agree with H.264. Experimental results show that the proposed methods are quite effective in the performance improvement of the H.264 encoding. We also propose an automatic method for video segmentation which can find out the moving object in the video even when the video is taken by non-static camera. Experimental results show the correct segmentation rate are quite good.

致謝

在這兩年的求學過程中，要感謝曾經幫助過我、教導過我的人。首先謝謝指導教授莊仁輝老師，在兩年來的研究生涯中給予熱心的指導與教誨，老師常常在百忙之餘仍費心的指導我，因此使我的論文能夠更加的豐富與完整。同時，也感謝口試委員陳玲慧教授、雷欽隆教授和顏嗣鈞教授的建議和指教，使得本論文的內容更加充實。

在交大的這些日子，認識了許許多多新的朋友，讓我的求學生涯更加的快樂與順利，而實驗室裡的每一位同學、學長與學弟們也都很好相處，常常實驗室就像一個大家庭般的熱鬧、溫馨；也謝謝大家這些日子來的幫忙與指教，才能讓我在短短的兩年學到這麼多東西。

目錄

第一章 簡介.....1
第二章 H.264/AVC 視訊壓縮標準.....3
2.1 簡介.....3
2.2 預測編碼（Prediction Coding）.....4
2.2.1 畫面內的預測（Intra Prediction）.....4
2.2.2 畫面間的預測（Inter Prediction）.....5
2.3 轉換、量化與熵編碼.....10
2.3.1 轉換（Transform）.....10
2.3.2 量化（Quantization）.....10
2.3.3 熵編碼（Entropy coding）.....11
第三章 前景與背景之訊切割與壓縮的相關研究探討.....12
3.1 利用空間的資訊對靜態影像做切割.....12
3.2 利用時空的資訊對視訊影像做切割.....13
3.3 視訊壓縮的相關探討.....14
第四章 H.264 壓縮工具之分析、壓縮效能.....15
4.1 RDO（Rate-Distortion Optimization）壓縮效能分析.....15
4.1.1 RDO 參數λ與畫面品質、資料量之關係.....16
4.1.2 RDO 參數λ與區塊大小之關係.....19
4.2 使用前景與背景提升壓縮效能.....25
4.2.1 前景與背景的使用觀念和流程.....25
4.2.2 前景畫面品質之提升.....26
4.2.3 背景資料量之減少.....30
4.2.3 前景與背景壓縮效能展示.....31
第五章 視訊影片中運動物體之自動切割.....34
5.1 視訊切割概念與流程.....36
5.2 區域運動估測與全域運動估測.....37
5.2.1 區域運動估測使用區塊比對.....38
5.2.2 全域運動估測使用 PTZ 攝影機運動模型.....39
5.3 自動區域運動估測與手動全域運動估測.....44
5.4 自動區域運動估測與半自動全域運動估測.....49
5.5 自動區域運動估測與自動全域運動估測.....52
5.6 結合小區域雜訊之去除之自動視訊切割方法.....59
5.7 不同參數設定的影響與實驗結果.....60
5.7.1 運動向量差異 d_{motion} 之臨界值選取.....60
5.7.2 參考畫面 I_{t+1} 的選取與切割結果之關係.....60
第六章 結論與未來展望.....70

表目錄

表 1：I、P 與 B 畫面所可以使用到的巨區塊（區塊）的種類 9
表 2：QP 量化對應值 11
表 3：平均 Frame Size 與 PSNR 之比較（Foreman） 18
表 4：RDO 不同參數之壓縮時間（Foreman 影片 11 張畫面） 19
表 5：影片壓縮時間比較（59 張畫面） 32
表 6：壓縮資料量與 PSNR 之比較（59 張畫面） 33
表 7：圖 26 (a) 與圖 26 (b) 的攝影機參數 41
表 8：訓練群終點誤差分佈（共 60 對） 41
表 9：測試群終點誤差分佈（共 30 對） 41
表 10：終點之誤差統計（共 450 對） 41
表 11：圖 27 (a) 與圖 27 (b) 所求得的終點誤差之統計(共 44 對) 42
表 12：圖 28 (a) 與圖 28 (b) 的攝影機參數 44
表 13：圖 28 (a) 與圖 28 (b) 所求得的終點誤差之統計(共 35 對) 44
表 14：半自動全域運動估測計算出的攝影機參數 50
表 15：修正的偏差參數 56
表 16：使用不同參考畫面時的切割正確率（共切割 50 張畫面） 68
表 17：切割時間 69

圖目錄

圖 1：影片利用前景背景編碼的示意圖.....1
圖 2：編碼流程示意圖.....3
圖 3：Macroblock:16×16、4×4，2 種區塊大小.....5
圖 4：I4×4 的 9 種預測模式[3].....5
圖 5：I16×16 的 4 種預測模式[3].....5
圖 6：巨區塊的七種區塊尺寸.....6
圖 7：切割畫面示意圖[3].....7
圖 8：Direct Mode 示意圖[7].....8
圖 9：區塊選擇流程圖.....9
圖 10：H.264 之整數離散餘弦轉換[3].....10
圖 11：不同λ對壓縮資料量大小的影響（Foreman）.....17
圖 12：不同λ對畫面的品質的影響（Foreman）.....18
圖 13：測試影片 Stefan、Foreman、Dancer.....20
圖 14：I-Frame 區塊大小分佈.....22
圖 15：P-Frame 區塊大小分佈.....23
圖 16：B-Frame 區塊大小分佈.....24
圖 17：不同λ對應畫面之區塊切割（Foreman）.....25
圖 18：應用前景和背景編碼示意圖.....26
圖 19：Foreman 原始影像與前景和背景的切割圖.....27
圖 20：Foreman 影片前景部分的比較.....28
圖 21：Foreman 影片背景部分的比較.....29
圖 22：Mobile 影片中的兩張畫面.....35
圖 23：運動向量示意圖.....35
圖 24：視訊切割架構.....36
圖 25：運動向量之差異性.....37
圖 26：Zoom 影片中的手動標示點.....40
圖 27：Pan、Tilt、Zoom 影片中的手動標示點.....42
圖 28：Mobile 影片中的手動標示點.....43
圖 29：全域運動估測流程圖.....44
圖 30：自動區域運動估測與手動全域運動估測之視訊切割架構圖.....45
圖 31：區域運動向量與全域運動向量.....46
圖 32：16×16 區塊之區域運動向量與全域運動向量.....47
圖 33：Mobile 畫面之自動區域運動估測與手動全域運動估測 d_{motion} 值.....48
圖 34：Mobile 畫面切割結果之一.....48
圖 35：自動區域運動估測與半自動全域運動估測之視訊切割架構圖.....50
圖 36：自動區域運動向量與半自動全域運動向量.....51

圖 37：Mobile 畫面之自動區域運動估測與半自全域運動估測 d_{motion} 值.....51
圖 38：Mobile 畫面切割結果之二.....51
圖 39：自動視訊切割架構圖.....52
圖 40：全域運動估計與偏差參數修正流程圖.....54
圖 41：修正偏差參數所使用的對應點.....55
圖 42：Mobile 畫面之自動區域運動估測與自全域運動估測 d_{motion} 值.....57
圖 43：Mobile 畫面切割結果之三.....57
圖 44：欲切割的 PTZ 影片.....58
圖 45：PTZ 影片切割結果.....58
圖 46：圖 44 使用全自動切割方法之 d_{motion} 值.....58
圖 47：去除雜訊之切割.....59
圖 48：Mobile 影片中的六張影像.....62
圖 49：使用不同參考畫面所求得的區域運動向量與全域運動向量.....64
圖 50：使用不同參考畫面所求得的 d_{motion} 分佈.....66
圖 51：使用不同參考畫面之自動切割結果.....67

第一章 簡介

由於科技的進步與資訊產業的蓬勃發展，多媒體應用日漸廣泛已成為生活中不可以缺少的一部分，一般的影音資料如果未經壓縮，資料量將會非常龐大而難以儲存與傳送，許多壓縮方法因此因應而生。針對視訊壓縮方面，H.264/AVC [1][2][3][4]是 2003 年所被提出的視訊壓縮標準，考慮使用者的需求與壓縮上的彈性，本論文提出一個前景與背景的分析應用在現有 H.264/AVC 的壓縮標準上，使得整體壓縮時間減少而著重部分畫面品質提升，並且有助於壓縮資料的儲存、傳送與利用。

由 ISO Motion Picture Experts Group（MPEG 組織）和 ITU-T Video Coding Experts Group（VCEG 組織）所成立的 Joint Video Team（JVT 組織），提出的 H.264/AVC（Advanced Video Coding）（或 MPEG4 Part 10/AVC）標準，為本論文所要探討的對象。本論文將針對現行的視訊壓縮標準 H.264，探討前景和背景的應用，目的在透過前景的影像品質的提升、背景的資料量之降低，以及壓縮速度之提升等三方面，來改善整體標準的壓縮效率。

對於本論文所提出的視訊壓縮方法，使用者可以依據需求分別定義前景與背景，一般而言，前景是影片中重要的部分，可能是物體移動的部分，或是觀看者仔細注意的畫面部分；而背景可能以靜態區域為主，或是整段影片中觀看者較不注意的部分。

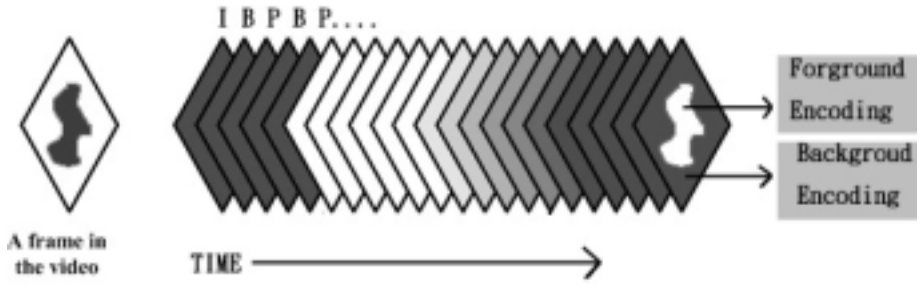


圖 1：影片利用前景背景編碼的示意圖

對於整個視訊壓，其過程分為編碼流程（Encoding Process）和解碼流程（Decoding Process）2 部分，分別用編碼器（Encoder）和解碼器 （Decoder）來實作，而本論文所討論的對象是編碼過程。圖 1 為我們如何利用前景和背景來編碼的示意圖，其中是把影片的內容，區分為 2 個部分後，在編碼器中以不同的編碼流程來處理，達到不同的編碼效能和編碼品質。

本論文中所提出的編碼方法可以使用切割好的前景與背景遮罩作為輔助的壓縮資訊，另外，本論文也提出一套自動視訊切割方法可供使用者使用。對於前景與背景的分析，分別找出適合前景與背景的區塊大小與 Rate-Distortion Optimization [5][6][7]參數，來提升前景影像品質、降低背景資料量，並減少壓縮速度。視訊切割方面則是利用物體的運動資訊，切割出與攝影機運動不一致的物體。

經過本論文所提出的方法壓縮後的影片，皆可以使用在一般的 H.264 解碼器上，不會有相容性的問題，因為本論文的壓縮方法皆符合 H.264 標準所定義的語法（Syntax）和語意（Semantics），而本論文所考量壓縮效能的依據有：訊號雜訊比（SNR, Signal to Noise Ratio）、位元速率（Bitrate）、編碼速度（Coding Speed）…等。

本論文共分成六章，第二章是 H.264 視訊壓縮標準及其壓縮原理的簡介。第三章是“前景和背景的應用”與“視訊切割”之相關研究討論。第四章討論本論文所提出的“前景背景應用方法”與實驗結果。第五章為本論文所提出的自動視訊切割方法與實驗結果。第六章為結論，並且探討一些將來可能繼續研究的方向和應用。

第二章 H.264/AVC 視訊壓縮標準

2.1 簡介

MPEG 的全名為“ Motion Picture Expert Group”，是由 IEC/ISO 在 1988 年成立的組織，主要的工作是制定標準化的數位視訊壓縮演算法。它分別在 1991、1994、1998 年了制定 MPEG-1 [8]、MPEG-2 [9]、MPEG-4 [10]等視訊壓縮標準。

對於資料的日漸龐大，相較於其他的視訊壓縮標準，ITU-T Video Coding Experts Group（VCEG）希望建立一個效率加倍的標準，而提出了 H.26L。2001 年，VCEG 和 MPEG 共同合作組織了 Joint Video Team（JVT）並且在 2003 完成了整個新的視訊壓縮標準 H.264/AVC（或稱 MPEG-4 Part10 或是 MPEG-4 /AVC）。

在 MPEG 壓縮標準中，只定義了視訊資料經壓縮後的位元串列的語法和語意，以及解碼的程序；對於編碼的過程，在標準內並沒有明確的制定，主要可提供編碼器設計者根據編碼速度、品質和複雜度來做取捨，只要編碼後的位元串列符合 MPEG 標準定義的語法和語義即可。

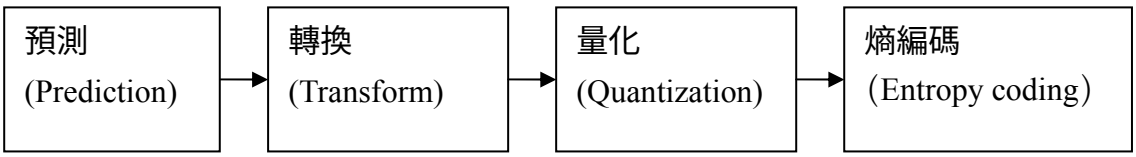


圖 2：編碼流程示意圖

現今的視訊壓縮標準大多採用圖 2 所示的編碼流程，在編碼的過程中，一個巨區塊（Macroblock）為編碼的基本單位，一個巨區塊大小為 16×16 像素，每一個畫面在壓縮時都會被區分成為 $M \times N$ 個巨區塊（M 與 N 為正整數），而在 H.264 中每一個巨區塊都可能再細分成數個區塊（Block）。有了以上的基本觀念，我們將在接下來的小節中，依序對 H.264 的預測、轉換、量化、熵編碼等步驟做介紹。

2.2 預測編碼（Prediction Coding）

視訊影片通常具有空間與時間上的相似性，此為視訊壓縮中重要的概念。就空間上而言，相鄰的像素通常在色彩或灰階強度上會非常相似，此時就可以利用周圍已編碼過的像素來預測尚未編碼的像素；當然，在某些特別的情況中可能空間上的相似性不高（如物體的邊緣），但大多數情況是具有空間上相似的特性。由於影片是由多張靜態影像的連續播放，為了使人眼觀看感覺流暢，相鄰的二張畫面當中，大部份的場景也都有極高的相似性或不變性，連續兩張畫面中的物體可能只有些微的運動差異，利用這種時間上的相似性，只要知道物體的運動便可由上張畫面找到相似物體。利用空間相似性來預測稱為“畫面內的預測 ”（Intra Prediction），利用時間相似性的預測則稱為“畫面間的預測 ”（Inter Prediction）。

2.2.1 畫面內的預測（Intra Prediction）

使用 Intra Prediction 的畫面稱為 I 畫面（Intra-Coded Frame）。對於 I 畫面中的巨區塊，如圖 3 所示，H.264 可以直接使用 16×16 巨區塊壓縮或再切割成 16 個 4×4 區塊做壓縮；此兩種區塊又分別可稱為 I16×16（Intra 16×16）與 I4×4（Intra 4×4）區塊。利用空間上的相似性，I4×4 有九種預測模式可使用，I16×16 有 4 種預測模式可使用，分別如圖 4、圖 5 所示。編碼器會對每一種預測模式算出預測區塊的像素值，並計算預測區塊與真實區塊之誤差；之後，編碼器再挑選誤差最小之模式為最佳的預測模式。計算預測區塊的公式可在[4]找到，因為解碼器也可以藉由同樣的公式計算出預測區塊，所以需要傳送的部分只有最佳預測模式、區塊編碼種類（區塊形狀）與預測誤差值（或稱為 Residual）。

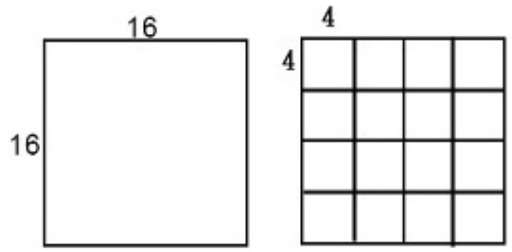


圖 3：Macroblock:16×16、4×4，2 種區塊大小

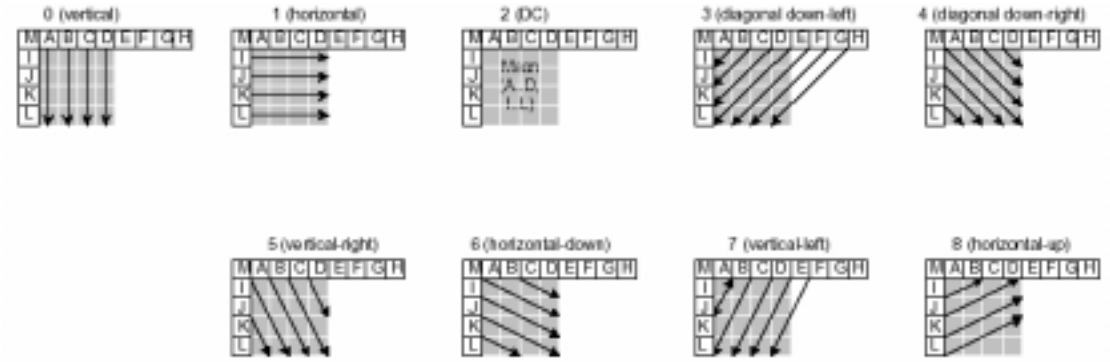


圖 4：I4×4 的 9 種預測模式[3]

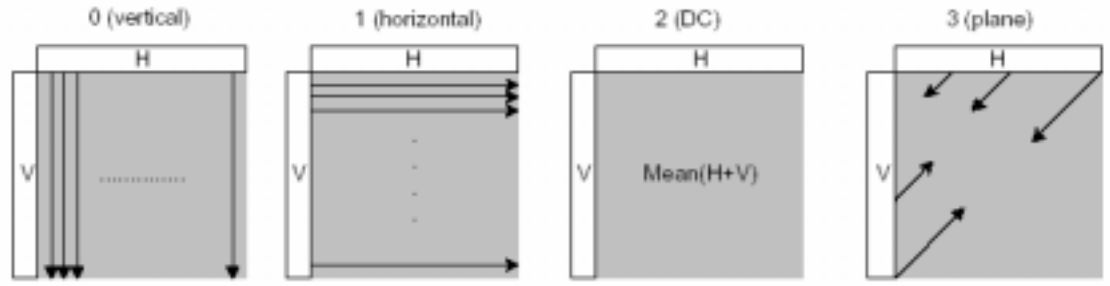


圖 5：I16×16 的 4 種預測模式[3]

2.2.2 畫面間的預測（Inter Prediction）

除了 I 畫面以外，有兩種畫面會使用 Inter Prediction 預測方式，分別稱為 P 畫面（Predictive-Coded Frame）與 B 畫面（Bidirectional-Coded Frame），其中的差異在於 P 畫面只能參考之前的畫面，而 B 畫面則可以參考之前與之後的畫面。對於目前要壓縮的巨區塊 (Current Macroblock)，編碼器會透過運動估測 (Motion

Estimation）找到參考畫面（Reference Frame）中最相似的區塊當作預測區塊。“目前要壓縮的區塊”與“預測區塊”之相對位移稱之為運動向量（Motion Vector）。類似於 Intra Prediction，編碼器只需要傳送運動向量與預測誤差，因為解碼器可以經由運動向量找到參考畫面中的預測區塊，再加上預測誤差就可以得到原本的真實區塊。在運動估測中通常會設定一個搜尋範圍，搜尋範圍越大可能會找到更相似區塊而減少預測誤差，但是搜尋時間也會因此會變大，而記錄運動向量所需要的資料量也可能會變大。

運動估測是以區塊比對為基礎，H.264 提供了七種不同尺寸的區塊作為運動估測之使用，並且每一種尺寸的區塊都需要記錄一個自己的運動向量。如圖 6 所示，一個巨區塊共可以有四種切割方式，分別可能為 16×16，8×16，16×8 或 8×8，當巨區塊被切割為 4 個 8×8 的區塊時，每個 8×8 區塊又可以有各自的四種切割方式。

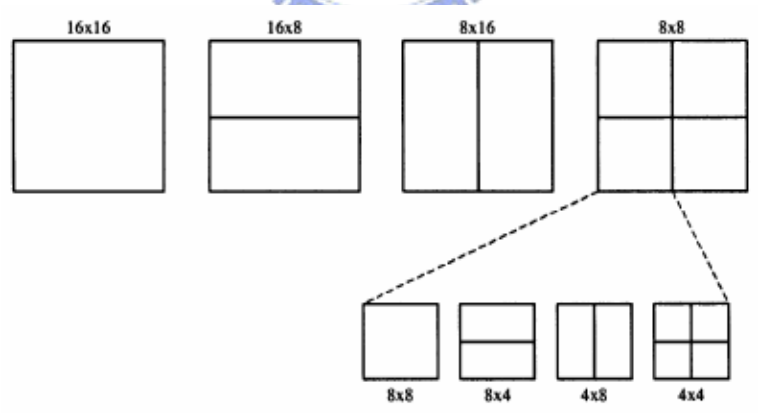


圖 6：巨區塊的七種區塊尺寸

因為區塊大小非固定，所以區塊的編碼種類也需要被傳送。如果選擇一種較大的區塊（如 16×16、16×8、8×16）來對巨區塊作切割，則巨區塊所需要的運動向量個數較少，因此運動向量資料量會變少，但預測誤差的資料量有可能會比

較多（因為誤差可能變大）。如選擇一個較小的區塊（如 4×8、8×4、4×4 等），雖然可以使得預測誤差資料量變小，但是因為區塊變小會增加的運動向量的總資料量，這些都是會影響到資料壓縮的效果。一般來說，較平滑的畫面通常用較大的區塊而較多細節部分的畫面則用較小的區塊會使的整體資料量較小，如圖 7 的影像即為一種可能的區塊組合樣式，較平滑的地方考慮使用較大的區塊，邊緣與輪廓等區域則可以考慮用小區塊組合而成。

除了上述的七種區塊大小外，P 畫面還有一種區塊模式，當一個巨區塊沒有編碼後剩下的預測誤差需要傳送，並且也不需要傳送運動向量的資訊時候，叫做 Skip Mode，此種類在解碼的過程中，並不使用到運動向量來表示預測後的結果，是直接複製參考畫面中對應位置之區塊來當作預測區塊。另外，在 B 畫面也有一種特別的區塊模式叫做 Direct Mode，此種模式會利用參考畫面的時間距離比率（Temporal Distance Scaling）來產生運動向量。

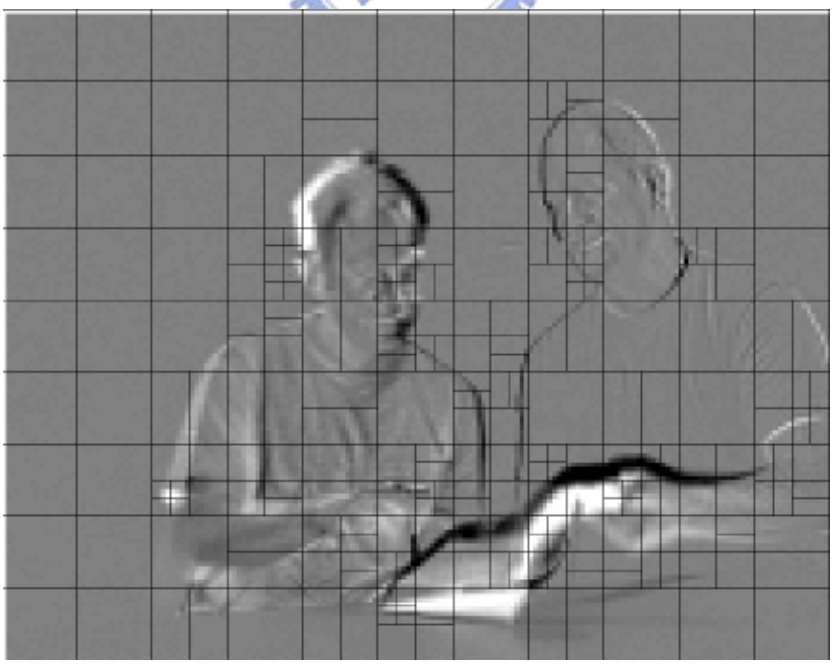


圖 7：切割畫面示意圖[3]

圖 8 為 Direct Mode 示意圖，（MV_0 , MV_1）分別表“目前要壓縮的區塊”之向前參考與向後參考運動向量。為了要求得這兩個運動向量，必須要由“向後參考畫面”（RL_1）中相同位置（Co-located）的巨區塊之運動向量“*MVc*”推導。若向前參考畫面為（RL_0），向後參考畫面為（RL_1），則二運動向量的求法為

$$MV_0 = （TD_B/TD_D）*MV_C \quad (2.1)$$

$$MV_1 = （TD_D - TD_B / TD_D）* MV_C \quad (2.2)$$

其中 TD_D 表示 RL_0 與 RL_1 在時間上的差值，TD_B 表示目前此 B 畫面與 RL_0 的差值。

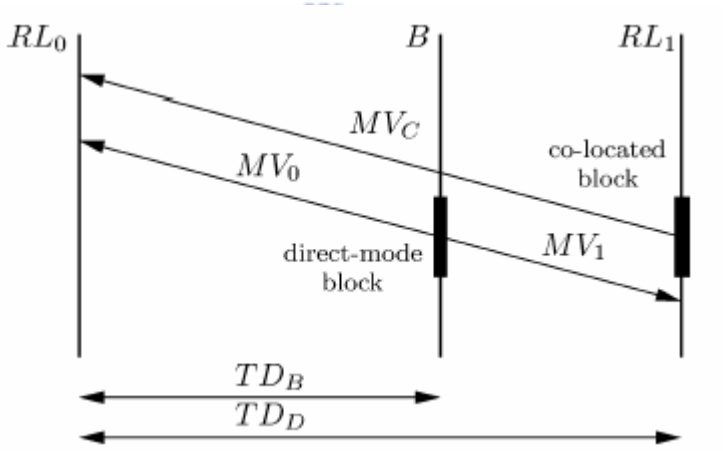


圖 8：Direct Mode 示意圖[7]

在 H.264 標準中，P 和 B 畫面，也可以使用 2.2.1 小節的 Intra Prediction 的 2 種區塊大小以及多種預測模式，當編碼器認為 Inter Prediction 預測方式之巨方塊誤差太大時便會使用 Intra Prediction 的預測方式。表 1 為 I、P 與 B 畫面所可以使用到的巨區塊（區塊）的種類之整理表列。圖 9 則為預測編碼時最佳區塊種類的選擇流程，其中何者為最佳的預測模式由編碼器的設計者決定，這是因為 H.264 標準只定義解碼器的規範，而 H.264 也提供了一種方法，這個方法可以在 H.264 原始碼 JM 版本[11]找到。

表 1：I、P 與 B 畫面所可以使用到的巨區塊（區塊）的種類

Frame Type	Skip	Direct	16×16	16×8	8×16	8×8	8×4	4×8	4×4	Intra16×16	Intra4×4
I										√	√
P	√		√	√	√	√	√	√	√	√	√
B		√	√	√	√	√	√	√	√	√	√

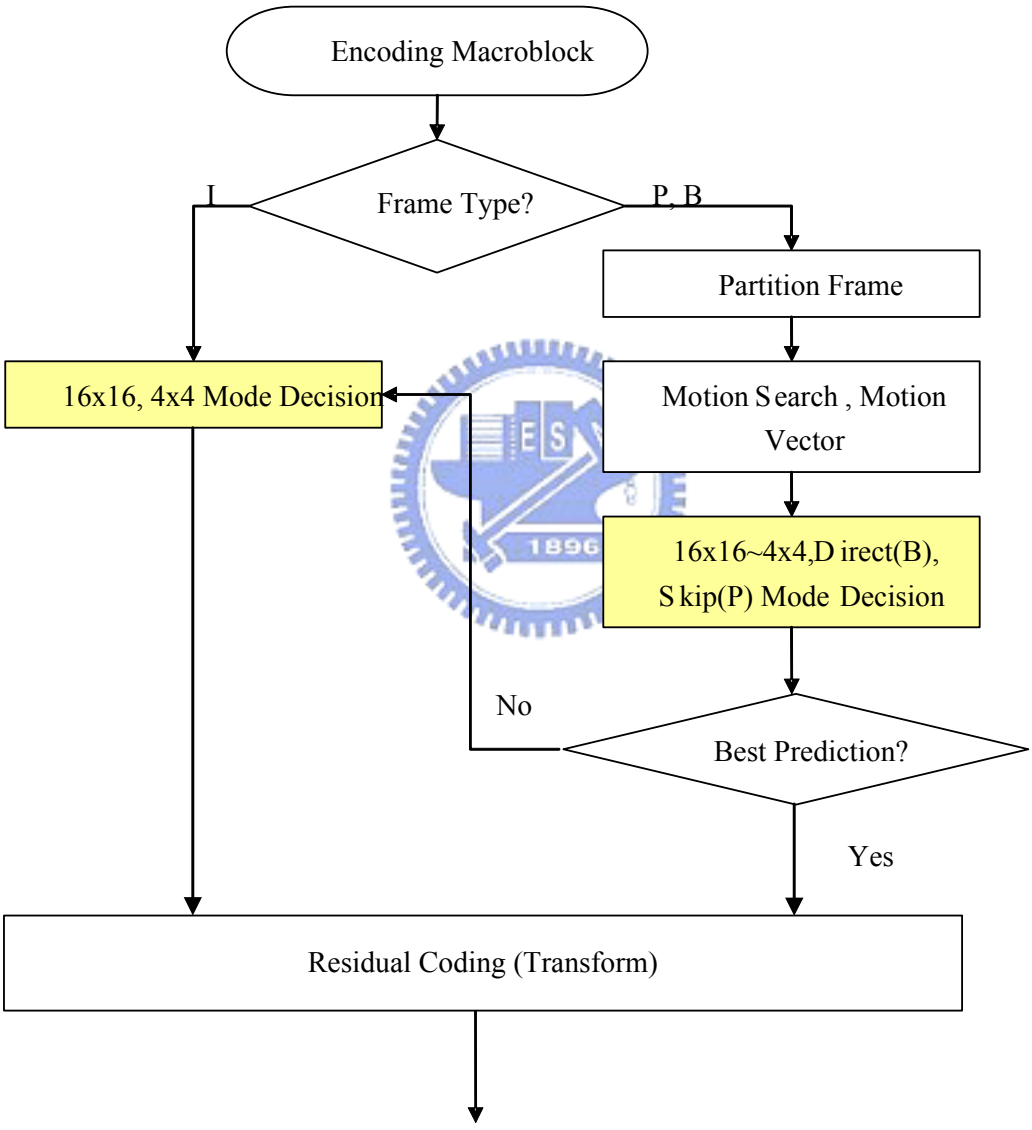


圖 9：區塊選擇流程圖

2.3 轉換、量化與熵編碼

2.3.1 轉換（Transform）

一般影像經過傅立葉轉換(Fourier Transform)或餘弦轉換(Cosine Transform)後，大部分的資料量都會集中在低頻的部分，高頻的部分則會留下較少的資料量，藉由這個的特性可以讓量化步驟更有效的處理資料。相較於之前的壓縮標準使用 8×8 區塊的浮點數離散餘弦轉換，H.264 則是使用了 4×4 區塊的整數離散餘弦轉換，使用較小的區塊減少了方塊與漣漪的現象，整數轉換則減少了浮點數轉換在電腦實作上誤差的累積。如圖 10，一個巨區塊可以被分成 16 個 4×4 區塊，每個區塊都會進行整數離散餘弦轉換，其中每個 4×4 區塊轉換後的 DC 成分即是原來區塊所有元素的平均值。在 H264 中新增了 Hadamard 轉換，H.264 會將 16 個 DC 成分集合成新的 1 個 4×4 區塊，對此 4×4 區塊進行 Hadamard 轉換進一步消除空間上的重複性，關於 H.264 中轉換方法的詳細介紹可以在[3][4]中找到。

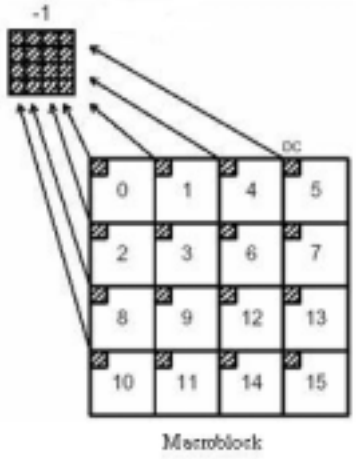


圖 10：H.264 之整數離散餘弦轉換[3]

2.3.2 量化（Quantization）

量化是一種失真的壓縮方法，資料經過量化之後即不可能恢復成原始資料，

表 2.2 為 H.264 所提供的量化參數（QP）表，每一個量化參數對應到一個量化位階（QStep），每當量化參數增加 6 時，量化位階呈 2 倍成長，而當量化參數越高時代表失真越嚴重，但是壓縮後的資料量也會較小。在這個步驟中所要量化的資料是經過整數離散餘弦轉換後的結果，當決定好量化參數之後，對每一個 4×4 區塊的量化參數就是固定的，但是在細節上，另外會有個權重調節低頻與高頻部分的量化比例，對於低頻的部分量化程度會比較低，失真度較小，高頻部分則量化程度較高，失真度較高。在前一小節提到，經過轉換後的高頻部分是對應到空間域上變化較快的細節，而人的視覺系統對空間域上變化較快的細節比較不敏感，所以在實際壓縮中容許的失真度可以較高，也因此量化程度較高；而低頻的部分若失真太嚴重人眼感覺較敏銳，因此量化程度必須較小，關於 H.264 中量化方法的詳細介紹可以在[3][4]中得到。

表 2：QP 量化對應值

QP	0	1	2	3	4	5	6	7	8	9	10	11	12	…
QStep	0.625	0.6875	0.8125	0.875	1	1.125	1.25	1.375	1.625	1.75	2	2.25	2.5	…
QP	…	18	…	24	…	30	…	36	…	42	…	48	…	51
QStep	…	5	…	10	…	20	…	40	…	80	…	160	…	224

2.3.3 熵編碼（Entropy coding）

視訊壓縮的最後一個步驟是熵編碼，H.264 中支援了 2 種熵編碼來進一步使得資料量變小，分別為 CAVLC（Context-Adaptive Variable-Length Coding）和 CABAC（Context-Adaptive Binary Arithmetic Coding）[3][4]，主要的概念是利用較少的位元來表示出現機率較高的符號，出現機率較低的符號則用較多的位元表示，如此一來整體的資料量較小，其中 CABAC 的方法複雜度較高壓縮效果也比較好。

第三章 前景與背景之訊切割與壓縮的相關研究探討

針對單一畫面中物體之切割，已經有許多方法被提出，這些方法可以根據不同的影像特徵（如顏色、紋理、灰階強度），切出符合這些特徵的物體，然而切割結果的穩定度相依於影像的特性而有好壞之分。在視訊檔案之中，單獨使用空間資訊所產生的切割結果略顯薄弱，如果可以加入時間域上的資訊（如運動資訊），便可增加切割的強固性。

3.1 利用空間的資訊對靜態影像做切割

一種常見的方法則是以區域為基礎（如 Region Growing）[12]，找出空間上特徵相同的區域，但此方法對於邊界的切割較不穩定，因為對於臨界值的選取亦較困難。為了切割出物體的輪廓，Snake Model[13]亦經常被使用。Snake Model 的缺點是必須要先以人工的方式，在物體的周圍圈出一個初始輪廓，然後根據 Snake Model 的內部力與外部力去逼近物體的輪廓。另外 Snake Model 也常常需要調整內部力與外部力的權重參數，以便適應不同特徵的影像。另外分水嶺演算法（Watershed algorithms）[14]亦經常被使用，分水嶺演算法是利用注水原理，從低處的像素開使用高處成長，而高處的地方通常是影像中的物體邊界，因此在物體邊緣處停止注水可以找到物體的切割。J. Malik 提出了使用 Normalized Cuts[15]來切割影像，主要的觀念是把一張畫面建構成一個加權圖，每個像素與像素之間會依照其相似關係而有一個權重。在切割的時候，一次的切割會切出兩群物體，其考慮的依據是去掉加權較小的邊並使同一群內的物體有最大的加權，如此一來比較相似的像素會被切割在一起，然而使用 Normalized Cuts 切割影像的時候，同一個物體有可能被切成幾個破碎的部分。

3.2 利用時空的資訊對視訊影像做切割

為了切割視訊影像，時域上的資料是非常有用的資訊，畫面中的物體運動是最常被拿來使用的。針對攝影機為固定的情況下，切割的方法會比攝影機在運動的情況簡單，最常使用的方法是連續兩張畫面的相減，用此可以偵測出畫面中有物體運動的區域，做為初步切割的結果。[16] [17]同樣是利用了兩張畫面間的差異對物體做切割，為了增加可靠性，[16] [17]會累多張積畫面的差異資訊再做切割。[18][19]推導了機率函數，將影片中相似的位置、顏色、運動等，判斷出相似的部分作群集，對不同特性的物體作切割。而[20]同樣是使用 Normalized Cuts 的方法，只是在建構加權圖時，考慮了運動的資訊做權重。

在攝影機不為靜止的狀況，對視訊畫面切割最為困難，[21]使用了物體追蹤與圖形識別的方法來切割運動的物體，此方法的好處是在物體有變形或者遮蔽時，仍然可已有不錯的切割效果，但是第一次步驟必須先用另外的方法做一個初始的切割，之後才可以使用物體追蹤的方式。[21]中並沒有提到一個強固的初始切割方法，而初始切割的好壞大大的影響了後續切割的結果，這一部分是仍需要改進的地方。[22]使用了仿射模型（Affine Model）來估測物體的運動，他的做法是每塊區域估計一組仿射參數，之後再合併仿射參數較像的鄰近區域。但是實際合併區域的時候，有可能屬於同一個物體卻不會被合併情況發生，因此造成同一個物體會被切割成許多的區塊。[23]同樣也利用了運動的資訊作為運動物體之切割，為了增加運動資訊的強固性，[23]所提出的方法必須要累積數張畫面間的運動資訊。接下來，求出具有不同運動之物體個數，並給每一個物體設定一組初始的仿射參數，來代表每一個物體。最後，推導出一個機率函數，函數所定義的是每一個像素屬於某個物體的機率，因此可以把每一個像素分類，對不同的物體作切割。[23]所提出的方法中，初始仿射參數的設定是一個重要的部分，另外，

具有不同運動的物體個數之求得也相當重要，由於這一方面的方法不穩定，造成有些的運動物體並沒有被切割出來。

3.3 視訊壓縮的相關探討

與壓縮效能相關的研究有可調式編碼（Rate Control Coding）[24]，在編碼時所考慮了的應用環境因素來調整編碼的效果，也就是彈性的使用多種編碼方式，再依據當時的一些條件，如: 頻寬、解析度等等的要求來進行編碼控制，這類方法著重在原來標準內的編碼工具的功能調整。另外在 MPEG4 包含了 Sprite Coding[25]的編碼技術，目的是對於每個畫面中重複出現的部分只做一次處理，進而減少重複編碼的情況產生，如此一來，便可以減少整體的壓縮時間與壓縮後的資料量，當每個畫面間的重複部分越大時，壓縮的資料量更會明顯的減少。然而在進行 Sprite Coding 時, 必須要有每張畫面切割好的遮罩當作輸入以做為輔助的編碼工具，如此壓縮的資料量與畫面的品質才會好，因此對於 Sprite Coding 的壓縮技巧上，視訊切割是一個重要的步驟。[26]提出了對 I 畫面的預測模式做改善，可以在畫面品質沒有大量減少與資料量沒有大量增加的情況下減少 I 畫面的壓縮時間；與[26]所提出的方法不同的是，本論文所提出的方法是針對 P、B 畫面的區塊選擇作改善，以減少壓縮的時間。

在本章，我們介紹了不同的切割方法與其在壓縮上的應用，本論文亦提出了一套視訊切割與壓縮的方法，並利用在現今視訊壓縮標準 H.264 上，其最後的目的希望能夠提升壓縮資料的訊號雜訊比（PSNR）、降低位元率（Biterate）與加快編碼速度（Coding Speed）。

第四章 H.264 壓縮工具之分析、壓縮效能之提升與前景背景之應用

在此章將介紹、分析 H.264 多重區塊尺寸與 Rate-Distortion Optimization (RDO)[5][6][7]等編碼工具對於壓縮效能之影響並將其應用在我們所提出的前景與背景壓縮方法。本章考慮的壓縮效能包含訊號雜訊比（PSNR）、位元率（Biterate）與編碼速度（Coding Speed）。

在第二章的介紹中得知，H.264 提供多種不同的區塊大小與多種預測方向的選擇，這些都是 H.264 比之前的壓縮標準在壓縮效能上大大提升的原因之一。另外 H.264 額外提供 RDO 來做為 Intra Prediction、Inter Prediction（Motion Estimation）的預測工具，其基本想法是想在影像品質與位元率之間取得一個平衡。

對於上述 H.264 壓縮工具， H.264 提供使用者在區塊大小上的選擇有相當大的彈性，RDO 編碼工具亦可以選擇是否加入編碼流程之中。

4.1 RDO（Rate-Distortion Optimization）壓縮效能分析

在 RDO 編碼過程中，所針對的 Rate 和 Distortion 的部份，是以下列方程式為主要依據：

$$Cost = D\left(B, \hat{B} \mid Q\right) + \lambda \cdot R\left(B \mid Q\right) \tag{4.1}$$

其中 *Cost* 為此巨區塊編碼方式的代價，Q 是給定的量化參數，B 是指原始影像中正在被處理的一個巨區塊，$\hat{B}$ 是原始巨區塊 B 經過編碼再解碼後的重建區塊，D 為估計失真度的函數，通常以 PSNR 當比較標準，R 為估計壓縮資料量的函數

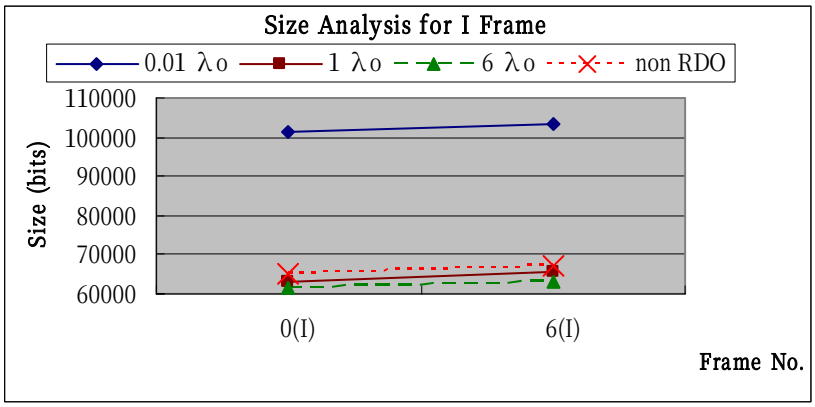
[5][6][7]，通常以 Bitrate 作衡量，λ 則是一個比例常數，藉由這個比例常數的調整，便可以對於畫面的品質和位元速率來做一個控制。當λ越小時，PSNR 值會越高，但壓縮率較低；而λ越大，壓縮率高，PSNR 值相對較低。

4.1.1 RDO 參數λ與畫面品質、資料量之關係

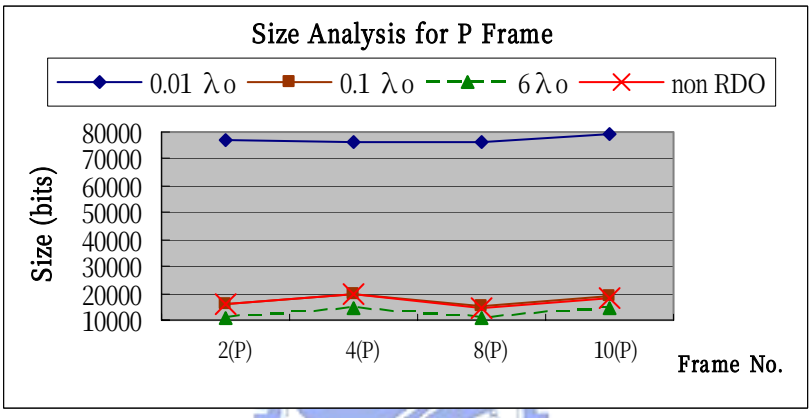
在這一小節，我們將驗證(4.1)式中 RDO 參數λ與畫面品質、位元率之關係。在 H.264 的參考原始碼中，預設的參數 I、P 畫面$\lambda_0 = 27.41$、B 畫面$\lambda_0 = 73.11$。我們以標準的$\lambda=1\lambda_0$來當作比較的基礎，此一數據將與$\lambda=0.01\lambda_0$、$\lambda=6\lambda_0$以及不使用 RDO 工具的壓縮方式相互比較。本章節實驗所做的一些設定所列如下：

- 影片 Foreman、壓縮 11 張畫面、序列格式 IBPBPBIBPBP
- I、P 畫面$\lambda_0 = 27.41$、B 畫面$\lambda_0 = 73.11$
- I、P、B 之 QP=28
- P、B 的參考畫面 1 張
- H.264 原始碼 JM 7.3[11]
- Windows XP Professional 、Visual C++ 6.0
- Pentum4-2.4GHz、256MB RAM

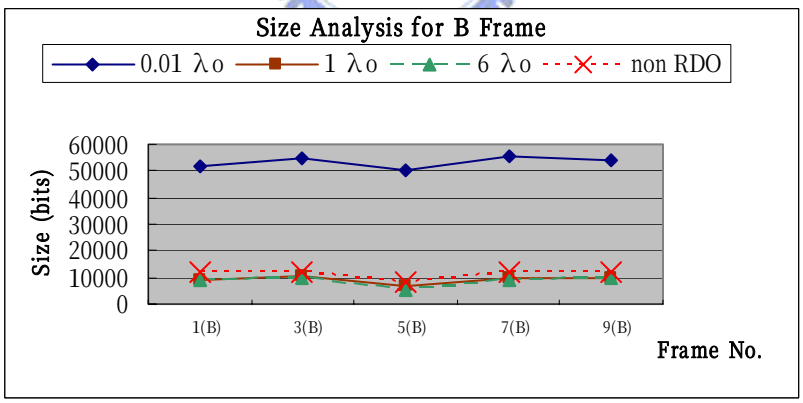
首先比較$\lambda=1\lambda_0$、$\lambda=0.01\lambda_0$、$\lambda=6\lambda_0$對壓縮效能的影響。圖 11 為不同λ對壓縮資料量大小的影響，圖 11（a）為 I 畫面之統計，圖 11（b）與圖 11（c）則分別為 P 畫面和 B 畫面之統計，I、P 與 B 畫面的共同結果是當λ越小時，壓縮率越低，λ越大則壓縮率越高。圖 12 為不同λ對畫面品質在 PSNR 上的影響，當λ越小時，PSNR 值會越高，λ越大，PSNR 越低。



(a) I Frame



(b) P Frame



(c) B Frame

圖 11：不同λ對壓縮資料量大小的影響（Foreman）

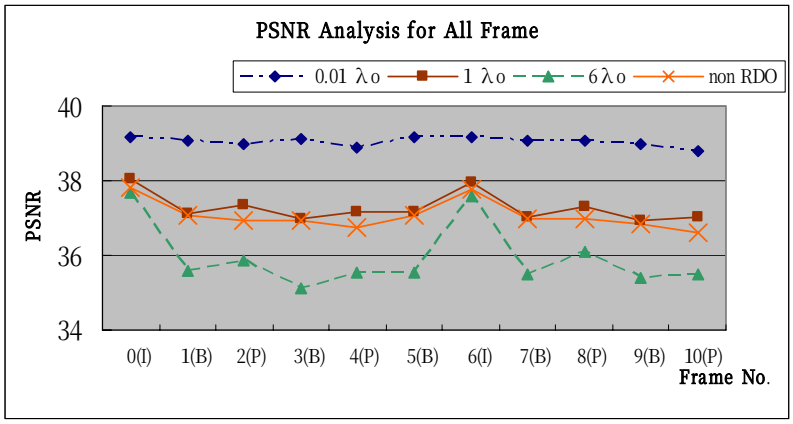


圖 12：不同λ對畫面的品質的影響（Foreman）

表 3：平均 Frame Size 與 PSNR 之比較（Foreman）

(a) 平均數據

	I Frame		P Frame		B Frame	
	Ave.Size(Bits)	Ave.PSNR(db)	Ave.Size(Bits)	Ave.PSNR(db)	Ave.Size(Bits)	Ave.PSNR(db)
$0.01\lambda_0$	102300	39.16	77118	39.06	53229	38.97
$1\lambda_0$	64200	37.59	16993	37.01	91889	37.36
$6\lambda_0$	62112	36.48	12614	35.40	847689	36.05
Non RDO	66272	37.42	17088	36.95	11189	36.99

(b) 以 $1\lambda_0$ 為基準之比較

	I Frame		P Frame		B Frame	
	Ave.Size(Bits)	Ave.PSNR(db)	Ave.Size(Bits)	Ave.PSNR(db)	Ave.Size(Bits)	Ave.PSNR(db)
$0.01\lambda_0$-$1\lambda_0$	38100(60.34%)	1.57	59681(342.28%)	2.05	44040(479.28%)	1.61
$6\lambda_0$-$1\lambda_0$	-2088(-3.25%)	-1.11	-4822(-27.65%)	-1.61	-712(-7.75%)	-1.31

表 3 為不同λ之數據比較，表 3（a）是使用不同的λ參數之平均數據，另外以 $1\lambda_0$ 為基準，將 $0.01\lambda_0$、$6\lambda_0$ 相對於 $1\lambda_0$ 的數據統計於表 3（b）中。在 I 畫面中，$0.01\lambda_0$ 比 $1\lambda_0$ 約提升了 1.57db 的 PSNR，但是多花了 60.34%的資料量。對 P 畫面，$0.01\lambda_0$ 比 $1\lambda_0$ 約提升了 2.05db 的 PSNR，但是多花了 342.28%的資料量。對 B 畫面，$0.01\lambda_0$ 比 $1\lambda_0$ 約提升了 1.62db 的 PSNR，但是多花了 479.28%的資料量；P、B 畫面雖然提高了較多的 PSNR 值，但是也花了更多倍的資料。

I 畫面中，$6\lambda_0$ 比 $1\lambda_0$ 約省了 3.25%的資料量，但是損失了 1.11db 的 PSNR。P 畫面中， $6\lambda_0$ 比 $1\lambda_0$ 約省了 27.65%的資料量，但是損失了 1.61db 的 PSNR。對 B 畫面，$6\lambda_0$ 比 $1\lambda_0$ 約省了 7.75%的資料量，但是損失了 1.31db 的 PSNR。P 畫面省下了最多資料量，但是損失的 PSNR 也是最多。

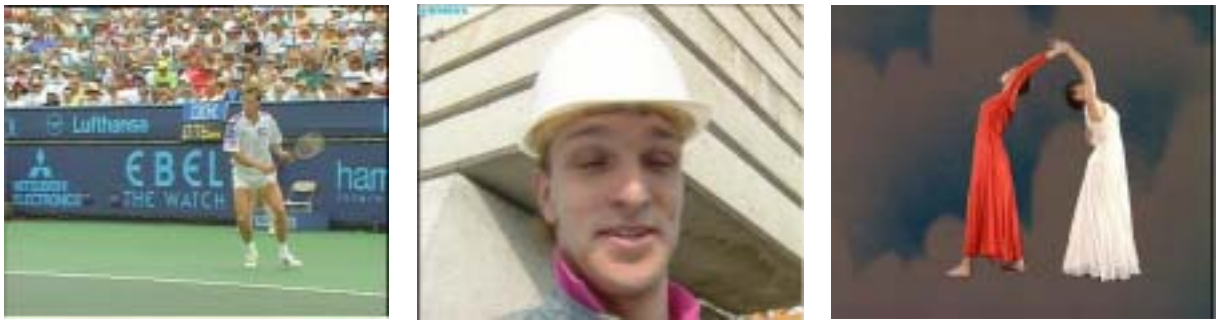
接下來我們比較使用預設的$\lambda=1\lambda_0$ 與不使用 RDO 之壓縮效能上的差異。由表 3（a）可知，在不使用 RDO 工具的情況下，所需要的資料量比 $1\lambda_0$ 的 RDO 略大，PSNR 則略為 $1\lambda_0$ 的 RDO 低，也就是說不使用 RDO 的情況，在壓縮率與失真度的表現上都較差；但是，RDO 為了達到更精準的估量，必須花費更多的計算時間，表 4 即為不同參數下，H.264 壓縮 11 張影片所要花費的時間，使用 RDO 工具大約要比不使用 RDO 工具多花上 2.5 倍以上的時間。

表 4：RDO 不同參數之壓縮時間（Foreman 影片 11 張畫面）

Coding Parameter	$0.01\lambda_0$	$1\lambda_0$	$6\lambda_0$	Non RDO
Coding Time (sec)	107.032	110.575	103.860	40.437

4.1.2 RDO 參數λ與區塊大小之關係

本小節將分析不同λ與所使用區塊大小之關係，我們使用了三種不同特性的影片當作測試，分別為 Stefan、Foreman、Dancer。Stefan 影片具有較多的細節區域，Dancer 影片大部分是非常平滑的區域，而 Foreman 特性則介於兩種影片之間，我們所使用的三種影片顯示在圖 13，至於本小節實驗的其他參數與環境相關設定則與 4.1.1 小節相同。



(a) Stefan (b) Foreman (c) Dancer

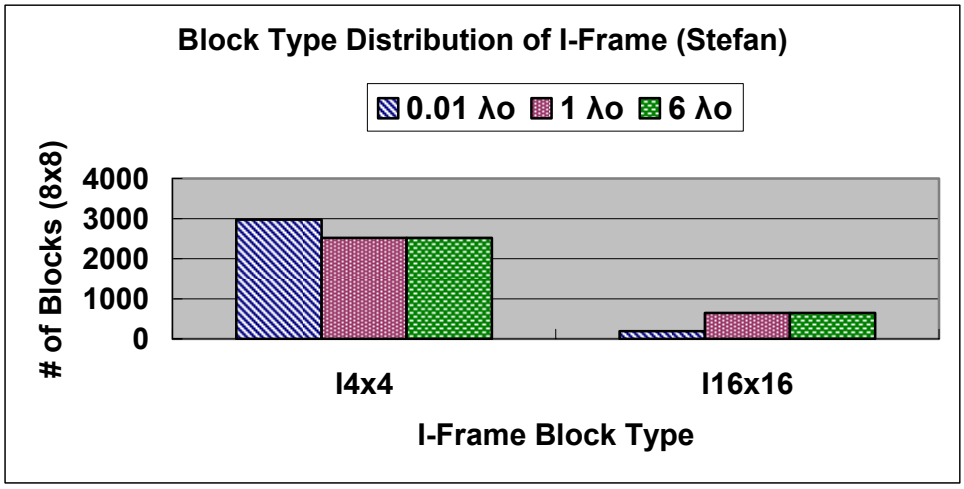
圖 13：測試影片 Stefan、Foreman、Dancer

圖 14 為我們針對 I 畫面所統計的結果，不同λ有不同的區塊大小分佈。$0.01\lambda_0$ 的 RDO 為了畫面品質之提升，比 $1\lambda_0$ 的 RDO 使用了更多的 I4×4 區塊；而 $6\lambda_0$ 的 RDO 為了壓縮率之提升，會比 $1\lambda_0$ 的 RDO 增加一點點 I16×16 區塊的用量。雖然 Stefan、Foreman、Dancer 三個測試影片在 $6\lambda_0$ 的 RDO 都增加了 I16×16 區塊的用量，但是在 Stefan、Foreman 整體使用 I4×4 區塊的比例仍然比 I16×16 區塊多很多。在第二章曾提到過，經過預測編碼完所要傳輸的資料包含：1.最佳預測模式、2.區塊編碼種類（區塊形狀）與 3.預測誤差值，其中區塊編碼種類所需要的資料量是固定的，而最佳預測模式與預測誤差則和區塊大小有很大的相關。區塊為 I4×4 時，因為區塊較小且預測模式較多（共九種），所以預測誤差會較小，但是相較於 I16×16 區塊只要傳送一個預測模式，16 個 I4×4 的區塊則要傳送 16 個表示預測模式的資料量（因為一個 16×16 區塊可以切割成 16 個 4×4 區塊）。因為 Dancer 影片平滑的區域相當多，使用 I16×16 區塊時之預測誤差已經相當小，所以不需要使用大量的 I4×4 區塊；而在 Stefan、Foreman 影片，$6\lambda_0$ 的 RDO 則選擇使用較多 I4×4 區塊使得壓縮資料量會較少。

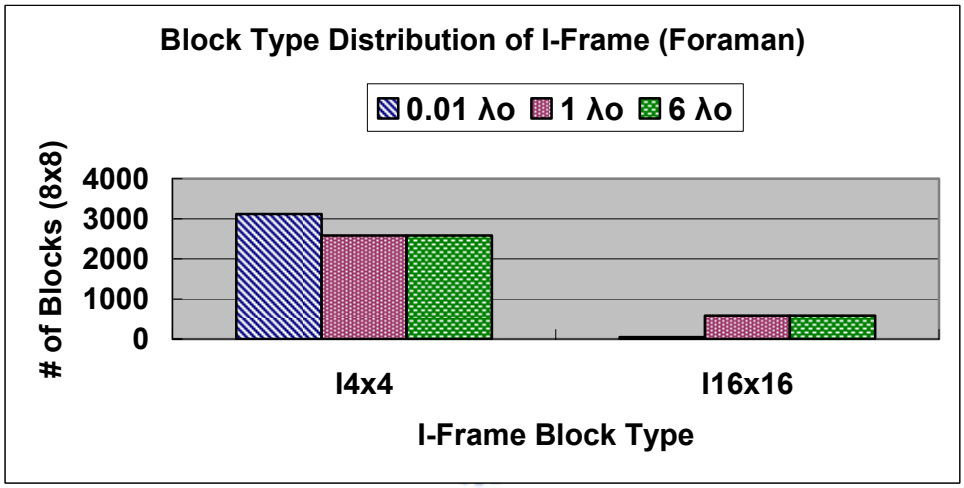
圖 15 是針對 P 畫面的情況所統計出不同λ對應的區塊大小分佈，$0.01\lambda_0$ 的 RDO 為了畫面品質之提升，比 $1\lambda_0$ 的 RDO 使用了更多的 4×4 與 Intra 區塊（I4×4 或 I16×16）；而 $6\lambda_0$ 的 RDO 為了壓縮率之提升，會比 $1\lambda_0$ 的 RDO 增加了更多 Skip Mode 的使用量，由圖 15 可得知，$6\lambda_0$ 的 RDO 之區塊使用量前兩名為 Skip Mode 與 16×16 區塊。圖 16 所討論的是 B 畫面，其中區塊分佈的關係大致上與 P 畫面類似，只是 B 畫面沒有 Skip Mode 區塊，取而代之的是 Direct Mode 區塊。

經由以上的實驗結果可以發現 P、B 畫面中，當λ越小畫面品質要求越高時，RDO 越傾向使用較小的區塊；當λ越大壓縮倍率要求越高，RDO 則傾向使用較大的區塊與特殊模式區塊。在 I 畫面中，當λ越小時，RDO 一樣傾向於使用較小的區塊，但是當λ越大壓縮倍率要求越高時，RDO 使用區塊尺寸的選擇則與影片的特性有很大的相關。而不使用 RDO 工具時，雖然其 PSNR 與位元壓縮率的分佈與 $1\lambda_0$ 的 RDO 相似，但其在 P、B 畫面區塊大小之選擇上，與 $1\lambda_0$ 的 RDO 還是存在著些許的差異；正因為在區塊大小上 $1\lambda_0$ 的 RDO 有做些細部的挑選，以致於 $1\lambda_0$ 的 RDO 在 PSNR 與位元壓縮率都有較好的結果。

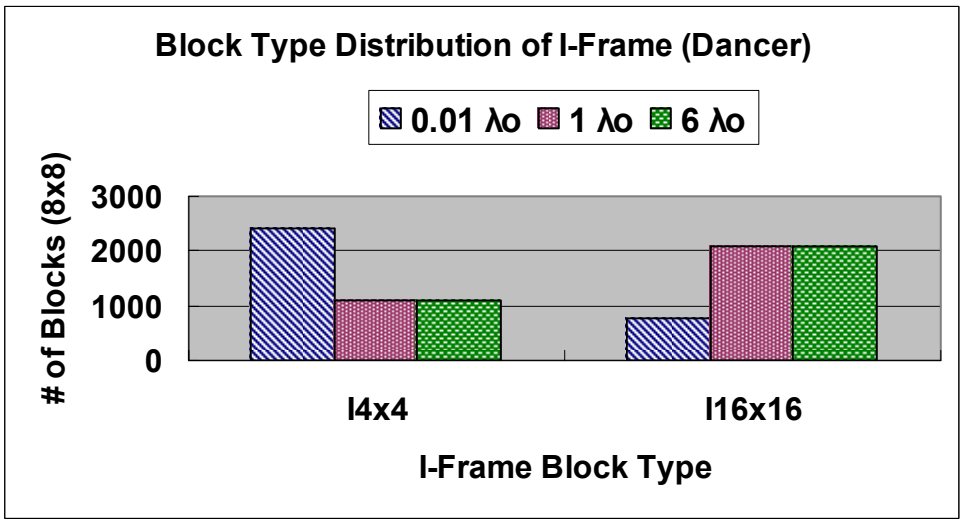
圖 17 為不同λ對應畫面之區塊切割，我們把實際區塊的選擇結果畫在影像上。此圖所顯示的是 Foreman 的第三張影像（Frame Number=2，Frame Type=P），圖中黑色的方格代表各式區塊的大小及形狀種類，灰色的為 Intra Prediction 的區塊，而紅色則為 P、B 畫面的 Skip 或是 Direct 區塊種類。從圖中更可以明顯的感覺到不同λ與區塊大小的關係，本小節的結果將會被下一小節所參考，並且加入前景與背景的使用來提升壓縮效能。



(a) Stefan

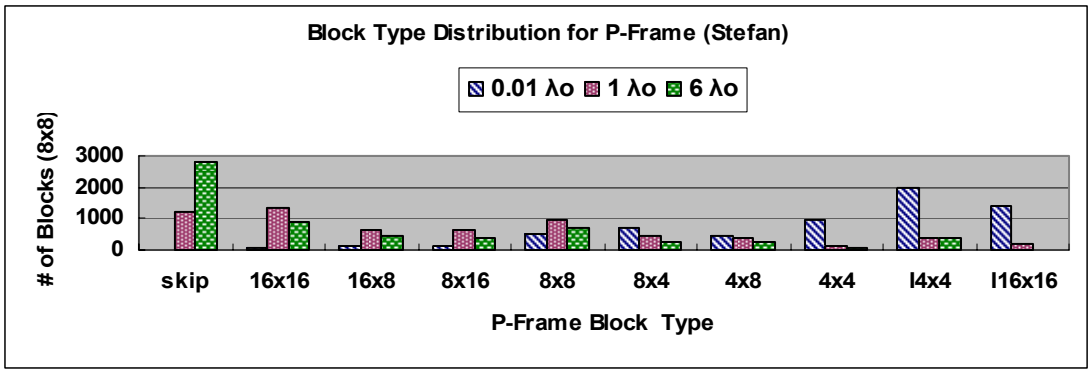


(b) Foreman

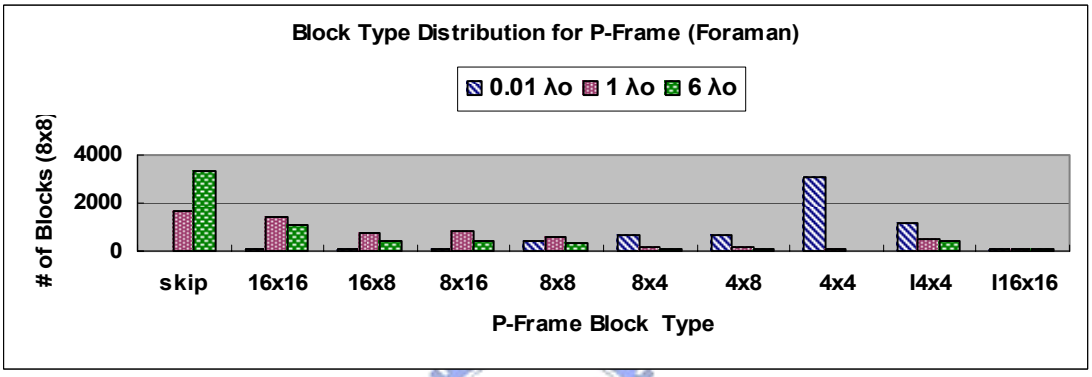


(c) Dancer

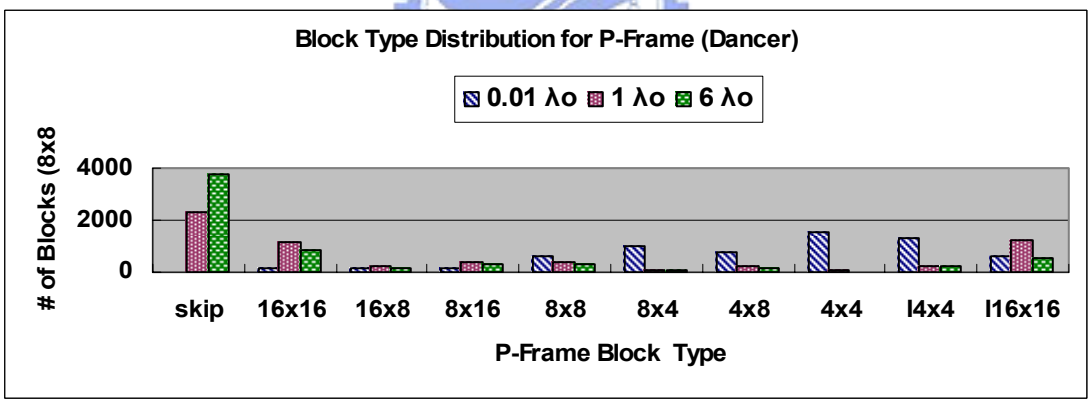
圖 14：I-Frame 區塊大小分佈



(a) Stefan

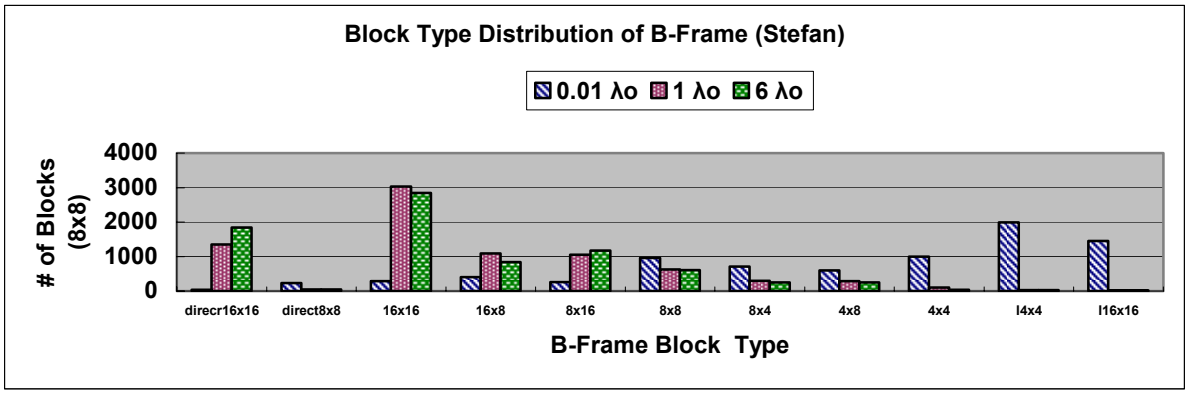


(b) Foreman

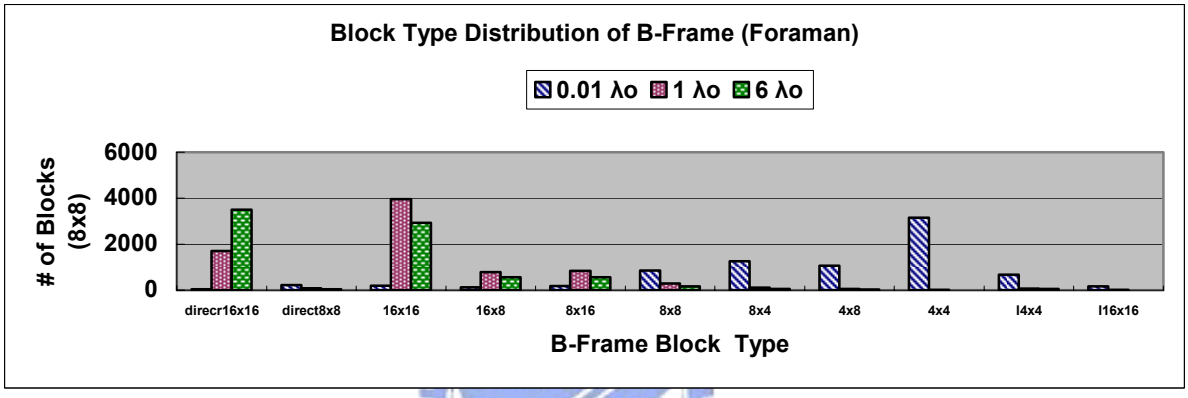


(c) Dancer

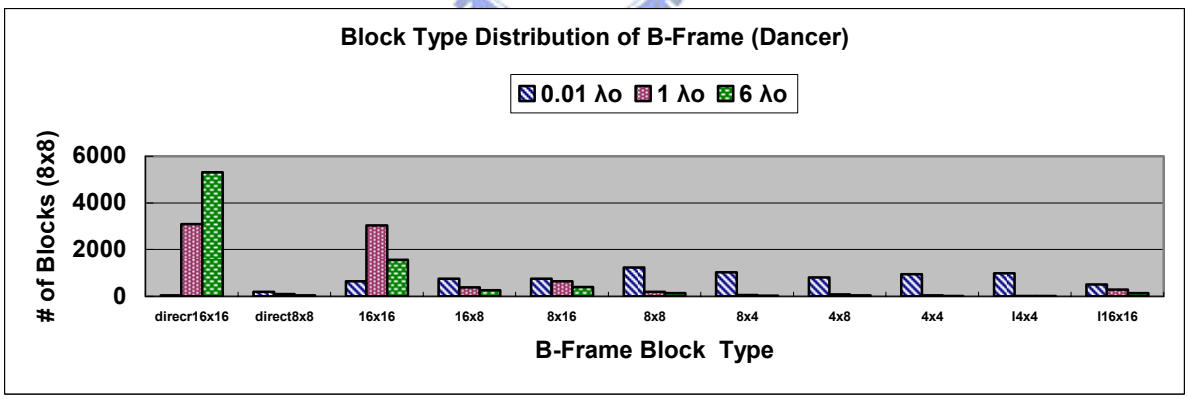
圖 15：P-Frame 區塊大小分佈



(a) Stefan

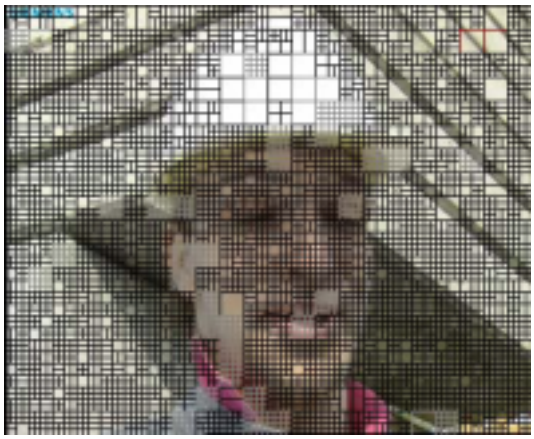


(b) Foreman



(c) Dancer

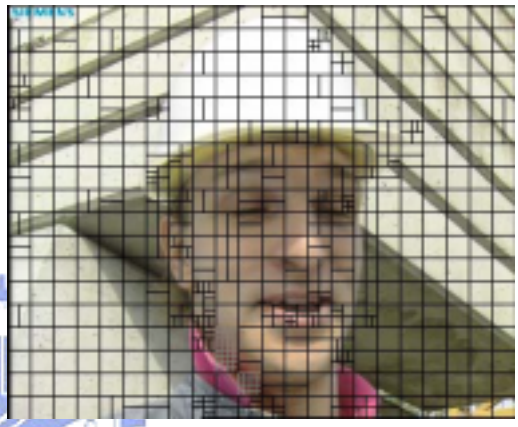
圖 16：B-Frame 區塊大小分佈



(a) 使用 $0.01\lambda_0$ 壓縮

(b) 使用 $1\lambda_0$ 壓縮

(c) 使用 $6\lambda_0$ 壓縮



(d) 不使用 RDO 壓縮

圖 17：不同λ對應畫面之區塊切割（Foreman）

4.2 使用前景與背景提升壓縮效能

這個章節，我們將應用已經切割好的前景與背景到編碼流程之中，針對不同的需求使用前景與背景，可以達成壓縮效能的提升。

4.2.1 前景與背景的使用觀念和流程

一般而言，觀看者在觀看一段影片的時候，可能只有某些事件的發生或動作會被觀看者特別注意，因此能有效的將這些區域區分出來，並加以用不同方式去壓縮，就可以達到重點區域的品質提升，而整體所耗費的資源下降。圖 18（a）

為一般視訊編碼方式，圖 18（b）中為我們所提出的想法，藉由原來影像與前景和背景分割的 2 元黑白影像，一起進入編碼器編碼，來改善壓縮效能。我們定義使用者特別注重畫面品質的部分為前景；相較於前景，比較不需要高畫面品質的部分定義為背景；對於背景，我們要求較高的壓縮倍率，即較少的資料量。為了符合前景與背景需求，在編碼的時候，前景與背景會使用不同的壓縮參數與方法編碼。

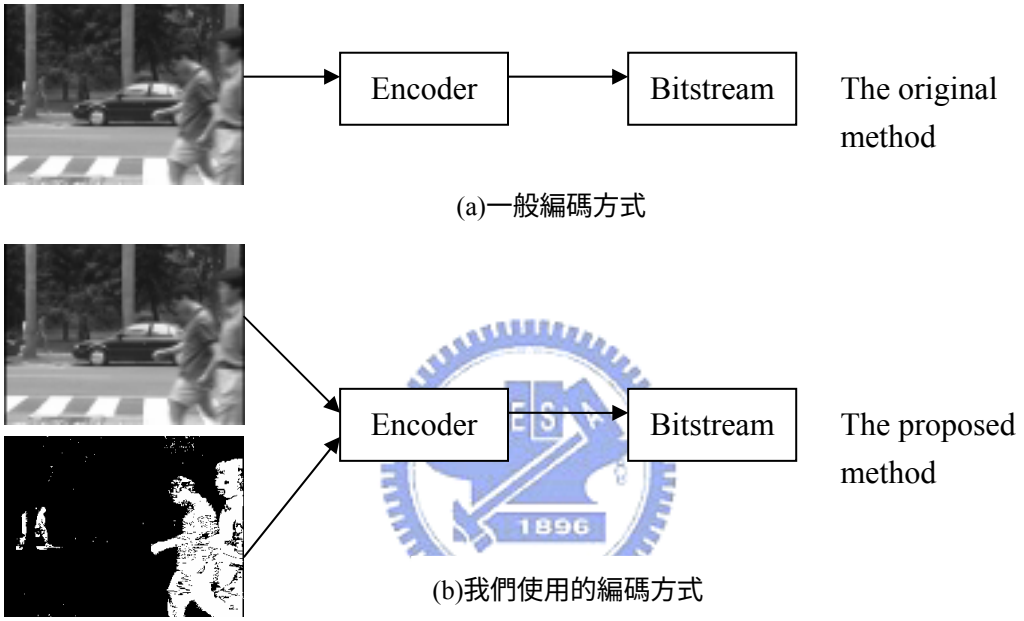
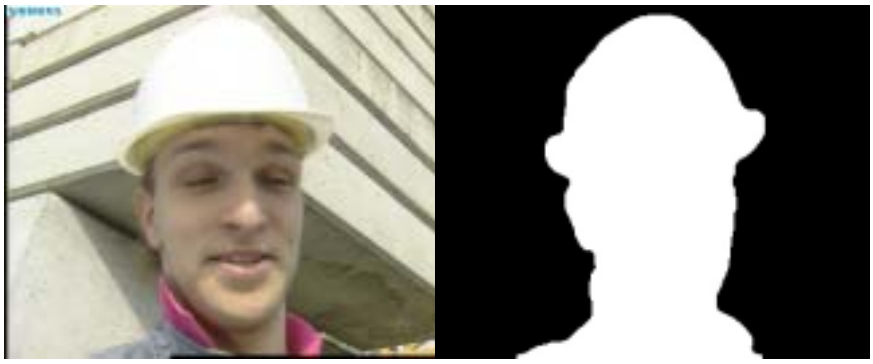


圖 18：應用前景和背景編碼示意圖

4.2.2 前景畫面品質之提升

在進行壓縮時，針對前景的部分，目的是容許資料量的增加而提升畫面的品質；本小節以 Foreman 影片作為實驗對象，針對圖 19（a）Foreman 的原始影像，我們加入了圖 19（b）前景與背景的切割圖作為輔助編碼的工具，並總共壓縮 59 張畫面，至於本小節實驗的其他參數與環境相關設定則與 4.1.1 小節相同。



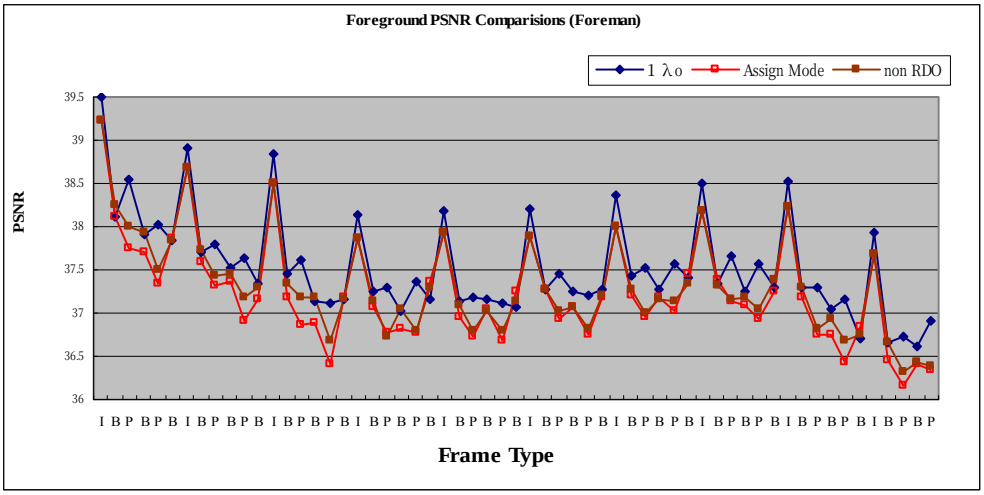
(a)原始影像　　(b)前景與背景的切割圖

圖 19：Foreman 原始影像與前景和背景的切割圖

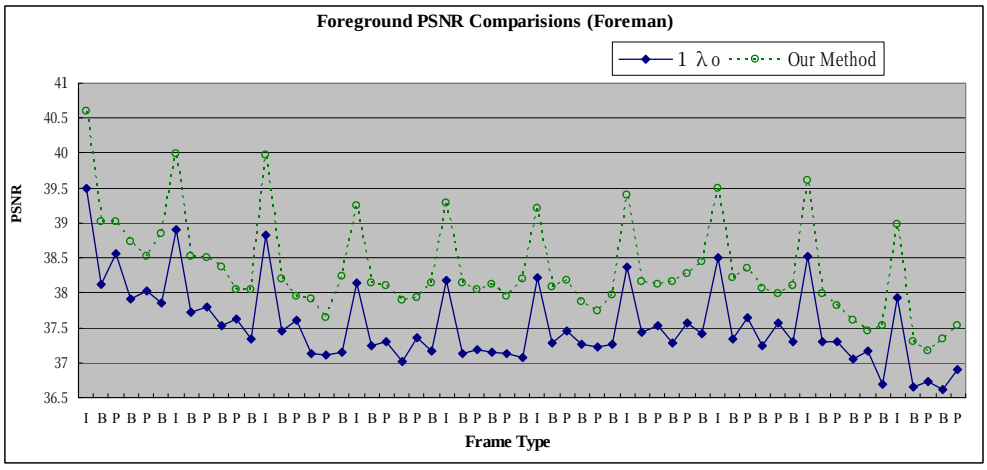
圖 19（b）中白色的區域代表所定義的前景，黑色的區域則定義為背景，因為 H.264 壓縮是以巨區塊為基本單位，所以一旦某個巨區塊裡包含白色的前景像素，則整個巨區塊皆會被當成前景區塊來壓縮，其餘不屬於前景的巨區塊則當成背景壓縮。經過 4.1.2 小節的分析，可知道在 P 與 B 畫面中，$0.01\lambda_0$ 的 RDO 為了畫面品質之提升，會大量使用 4×4、I4×4 與 I16×16 區塊，因此我們在不開啟 RDO 工具的情況下，指定 P 與 B 畫面前景的部分只能使用 4×4 區塊做編碼（不考慮特殊模式），而 I 畫面因為只有兩種區塊大小可以選擇，所以並不特別指定；此一方法稱做“指定模式”方法（Assign Mode），實驗的結果如圖 20（a）所示。由圖 20（a）可以發現，指定區塊的結果雖然可以和不使用 RDO 工具的效能相似，但還是比 $1\lambda_0$ 的 RDO 效能差。

在容許資料量增加而提升畫面的品質的前提下，我們提出了混和指定區塊與使用 RDO 工具的方法。我們指定 P 與 B 畫面前景的部分只能使用 4×4 的區塊，並同時使用 $0.1\lambda_0$ 的 RDO，而 I 畫面因為只有兩種區塊大小可選擇，所以並不特別指定區塊，只使用 $0.1\lambda_0$ 的 RDO 壓縮 I 畫面的前景，由圖 20（b）中可以看出我們的方法可以比 $1\lambda_0$ 的 RDO 大多提升 PSNR 0.5　1.25db。針對前景部分所使用的方法如下所列：

- 圖 20（a）所使用的 3 個方法：
  - 1. 使用 $1\lambda_0$ 的 RDO
  - 2. 不使用 RDO
  - 3. I 不指定，P 與 B 只能使用 4×4 區塊 （Assign Mode）
- 圖 20（b）所使用的 2 個方法：
  - 1．使用 $1\lambda_0$ 的 RDO
  - 2．我們的方法[27]：
    - I：只使用 $0.1\lambda_0$ 的 RDO
    - P、B：指定使用 4×4 區塊並使用 $0.1\lambda_0$ 的 RDO

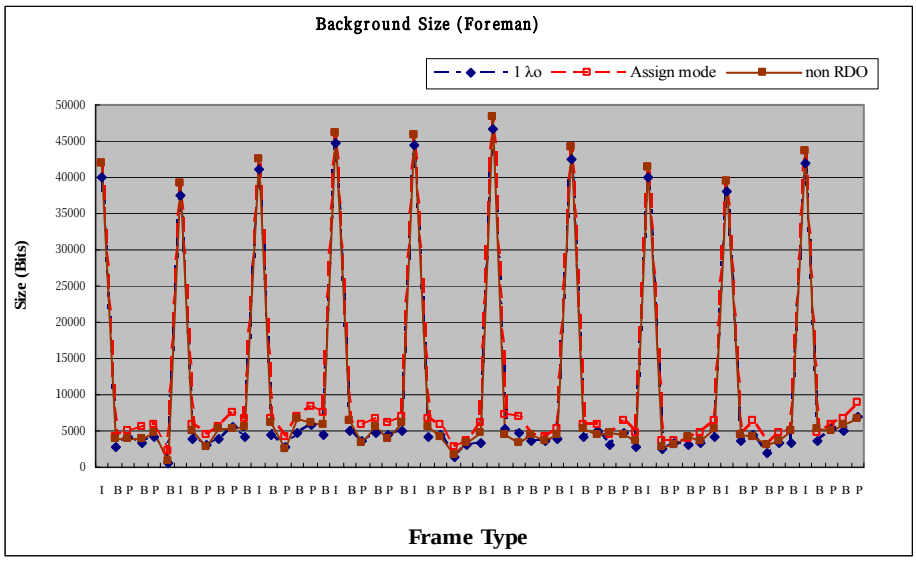


(a) PSNR-1

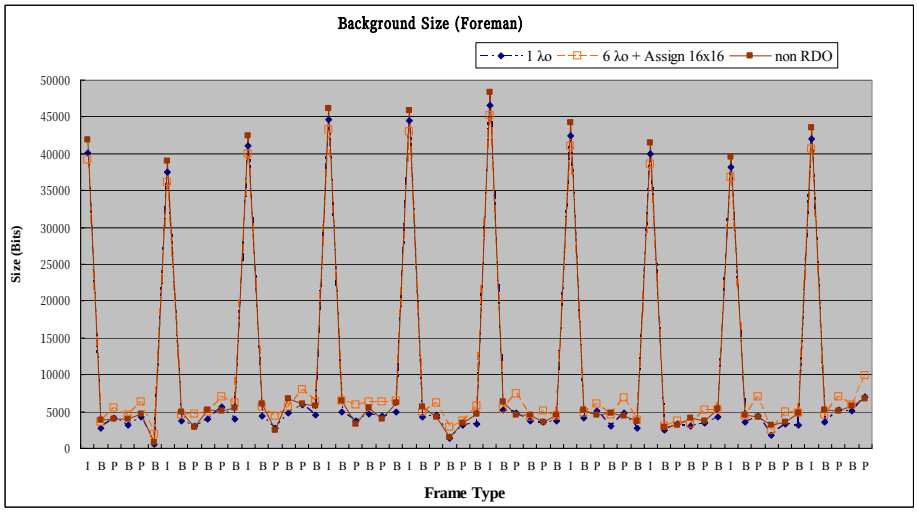


(b) PSNR-2

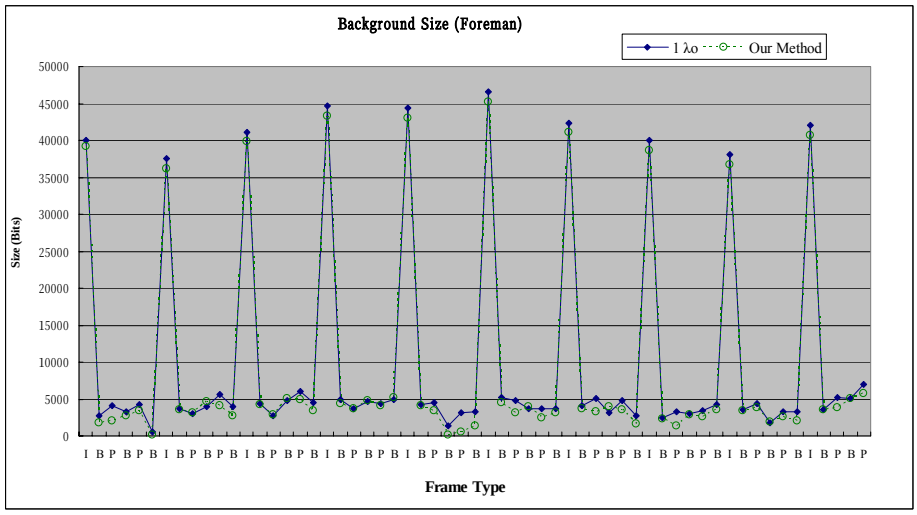
圖 20：Foreman 影片前景部分的比較



(a) Size-1



(b) Size-2



(c) Size-3

圖 21：Foreman 影片背景部分的比較

4.2.3 背景資料量之減少

在進行視訊壓縮時，背景是較不重要的部分，因此可以容忍畫面品質的犧牲，以達到減少資料量的目的。在 4.1.2 小節的分析，可以知道在 P 與 B 畫面中，$6\lambda_0$ 的 RDO 為了壓縮率之提升會大量使用 Skip Mode 與 16×16 區塊。對於 Foreman 影片，首先考慮 P 與 B 畫面之背景部分只能使用 16×16 的區塊，而 I 畫面不指定區塊，結果如圖 21（a）所示。圖 21（b）為 P 與 B 畫面之背景部分使用 16×16 的區塊，並同時使用 $6\lambda_0$ 的 RDO，而 I 畫面一樣不指定區塊，但是會使用 $6\lambda_0$ 的 RDO 壓縮 I 畫面的背景之結果。正如預期的，不管有沒有使用 $6\lambda_0$ 的 RDO，只固定使用 16×16 的區塊並不會減少很多的資料量，甚至比不使用 RDO 還差，因為壓縮後資料量是由運動向量與預測誤差的資料量總和所組成，16×16 的區塊雖然有較少的運動向量個數，但是預測誤差卻可能因此而增大。考慮資料量的減少，針對不同區域的影像特性使用適當的區塊大小才是最佳的壓縮方式。

我們進一步的改善這個方法，我們給予 $6\lambda_0$ 的 RDO 在背景的部分有兩種選擇，一種為 16×16 的區塊，另一種為 Skip 或 Direct Mode，結果如圖 21（c）所示，在大部分的情況下，我們可以比 $1\lambda_0$ 的 RDO 使用更少的資料量。針對背景部分所使用的方法如下所列：

- 圖 21（a）所使用的 3 個方法：
  - 1. 使用 $1\lambda_0$ 的 RDO
  - 2. 不使用 RDO
  - 3. I 不指定，P 與 B 只能使用 16×16 區塊（Assign Mode）

- 圖 21（b）所使用的 3 個方法：
  - 1. 使用 $1\lambda_0$ 的 RDO
  - 2. 不使用 RDO
  - 3. I：只使用 $6\lambda_0$ 的 RDO， P 與 B：指定使用 16×16 區塊與 $6\lambda_0$ 的 RDO ($6\lambda_0$＋Assign 16×16)
- 圖 21（c）所使用的 2 個方法：
  - 1．使用 $1\lambda_0$ 的 RDO
  - 2．我們的方法[27]：
    - I：只使用 $6\lambda_0$ 的 RDO
    - P、B：指定可使用 16×16 或 skip/direct mode 區塊並開啟 $6\lambda_0$ 的 RDO

4.2.3 前景與背景壓縮效能展示

從前兩小節的結果可得知，我們的方法可以針對不同的需求而改善壓縮效能，本小節將把前景與背景的資料同時呈現，並附上完整的實驗數據。

在壓縮影片所需要花費的時間上，不論是哪種測試影片，我們的方法與不使用 RDO 所花費的時間接近，使用 $1\lambda_0$ 的 RDO 之壓縮時間則為我們方法的 2.5 倍以上，實驗的結果顯示在表 5，由此可知我們的方法比單純只使用 RDO 工具節省了相當多的時間。

最後，壓縮效能數據（包含 Size 與 PSNR）呈現在表 6 中。關於前景之畫面品質，我們的方法在 Foreman、Stefan、Dancer 三個測試影片中比 $1\lambda_0$ 的 RDO

提升了 0.93db、0.32db、0.32db，同時也比不使用 RDO 工具好；然而，為了畫面品質之提升，我們的方法在 Foreman、Stefan、Dancer 三個測試影片中，前景部分會比 $1\lambda_0$ 的 RDO 多用了 2.3、1.4、1.6 倍的資料量。其中要注意的是，在同樣的參數同樣的方法下，提升 PSNR 值的程度與畫面特性有關。

表 5：影片壓縮時間比較（59 張畫面）

(a)Foreman

Coding Parameter	$1\lambda_0$	Our Method	Non RDO
Coding Time (sec)	553.192	218.113	204.516

(b)Stefan

Coding Parameter	$1\lambda_0$	Our Method	Non RDO
Coding Time (sec)	753.128	286.169	226.952

(c)Dancer

Coding Parameter	$1\lambda_0$	Our Method	Non RDO
Coding Time (sec)	484.390	189.649	144.156

在背景的部分，我們的方法在 Foreman、Stefan、Dancer 三個測試影片中，會比 $1\lambda_0$ 的 RDO 省下 7.1%、1.5%、10.8%的資料量，而耗損的 PSNR 值分別為 0.99db、1.88 db、2.4 db。至於整張畫面的總資料量就與前景背景所佔的比例、前影背景畫面的影像特性有關。

針對前景的 PSNR 提升與背景壓縮資料量的減少，我們的方法皆比標準 $1\lambda_0$ 的 RDO 好，其中整體所花費的資料量增加主要是使用在前景 PSNR 的提升，而我們的方法所花費的壓縮時間比標準 $1\lambda_0$ 的 RDO 少很多，與不開啟 RDO 的情況接近。

表 6：壓縮資料量與 PSNR 之比較（59 張畫面）

(a)Foreman

	Foreground of Frame		Background of Frame		Whole Frame
	Ave. PSNR (db)	Total Size (bits)	Ave. PSNR (db)	Total Size (bits)	Total Size (bits)
$1\lambda_0$	**37.52**	634315	36.29	**608598**	1174048
Our Method	**38.45**	1459147	35.30	**565450**	2024597
Non RDO	**37.30**	662309	36.47	**653861**	1316170

(b)Stefan

	Foreground of Frame		Background of Frame		Whole Frame
	Ave. PSNR (db)	Total Size (bits)	Ave. PSNR (db)	Total Size (bits)	Total Size (bits)
$1\lambda_0$	**34.67**	640358	35.80	**2658979**	3309337
Our Method	**34.99**	905867	33.92	**2617962**	3523829
Non RDO	**34.43**	670723	35.61	**2781574**	3452297

(c)Dancer

	Foreground of Frame		Background of Frame		Whole Frame
	Ave. PSNR (db)	Total Size (bits)	Ave. PSNR (db)	Total Size (bits)	Total Size (bits)
$1\lambda_0$	**35.60**	1062529	43.56	**192451**	1254980
Our Method	**35.92**	1697418	41.16	**171672**	1853411
Non RDO	**35.38**	1492036	43.07	**269959**	1761995

第五章 視訊影片中運動物體之自動切割

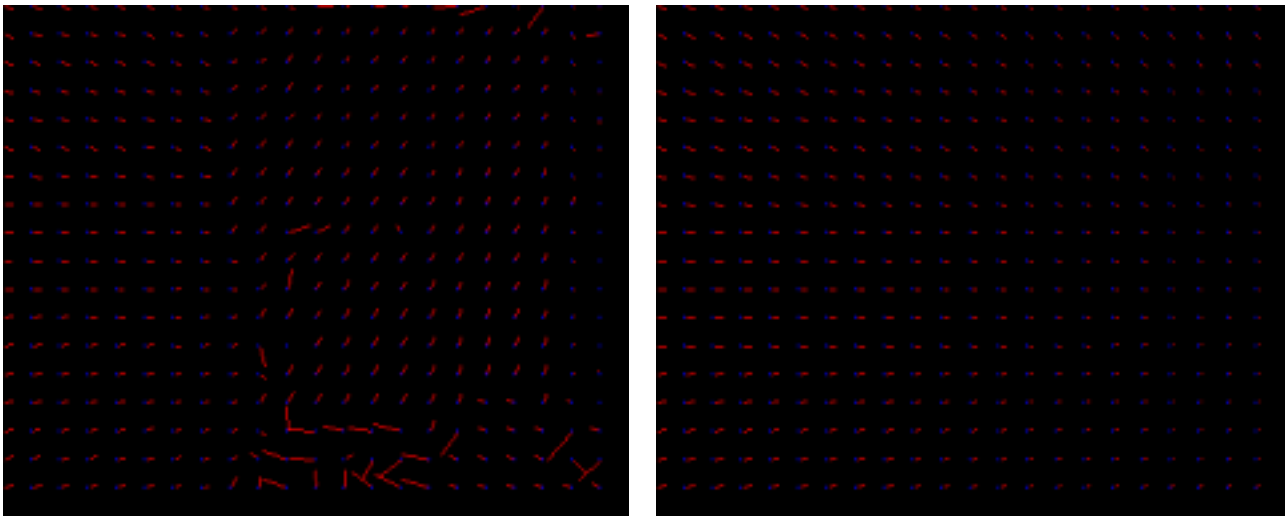
對於第四章所提出壓縮方法，除了要壓縮的影片外，另外需要輸入前景與背景遮罩來當作壓縮的輔助工具。在本章節，我們提出了一個視訊檔案的自動切割方法來產生前景與背景遮罩，在此，定義只因為攝影機之移動而運動的區域為背景（真實世界中靜止的物體），而除了攝影機運動外，尚有自身運動之物體則歸類為前景（真實世界中運動之物體），我們的方法可以在攝影機不為靜止的情況下，自動切割出真實世界中的運動物體。在我們所提出的方法裡，被切割的影像中之背景部分必須為最大的一群，如此才可以更準確的切割視訊檔案。

在攝影機不為靜止的影片中，如果有獨立運動的物體，則攝影機之運動與物體之運動會有其差異存在，若物體本身是靜止不動時，此時實際物體被攝影機拍攝的影像才會與攝影機運動一致。圖 22 為 MPEG 組織所提供的測試影片 Mobile，影片中獨立運動的物體有月曆、球與火車。假如圖 22（a）為目前要作切割處理的視訊畫面，而圖 22（b）為其鄰近時間點的參考畫面，則我們必須要求得圖 22（a）與圖 22（b）間各物體的運動資訊才能對視訊畫面做切割。圖 22（a）與圖 22（b）間的運動情況如圖 23（a）所示，可以看出圖 23（a）中的運動只有在部分的區域具有一致性，因為所拍攝的場景中有許多不同運動的物體存在。假設畫面中拍攝的所有物體均為靜止，則物體看起來會運動只是因為攝影機之運動造成，物體的運動趨勢（此時只有攝影機運動）可能如圖 23（b）所示。如果能夠找出圖 23（a）與圖 23（b）間運動的不同之處，就能分辨出何者為真實世界中靜止之物體，何者為運動之物體。

為了切割靜止與獨立運動的物體，我們利用了上述的觀念於我們的方法之中，做為視訊切割的基礎。本章所使用的方法是基於“攝影機運動估測模型”，在後續的小節會介紹視訊切割所需要的攝影機運動估測模型，並應用此模型到我們所提出的視訊切割方法中，進而達成視訊切割的目的。

(a) 處理畫面　　(b) 參考畫面

圖 22：Mobile 影片中的兩張畫面



(a)　　(b)

圖 23：運動向量示意圖

5.1 視訊切割概念與流程

視訊切割的流程如圖 24 所示，每次的輸入共兩個畫面，分別為“所要切割的畫面” I_t 與其鄰近的“參考畫面” I_{t+1} 。圖 24 演算法的第一個步驟是區域運動估測（Local Motion Estimation），區域運動估測是要發現兩張畫面間每個物體的位移。估測區域運動的方法有許多種，手動標示對應點即為一種較準確的方法，其他方法如所 2.2.2 小節所提到的區塊比對方法或光流估測（Optical Flow）[28] 皆為一種可行的方法。

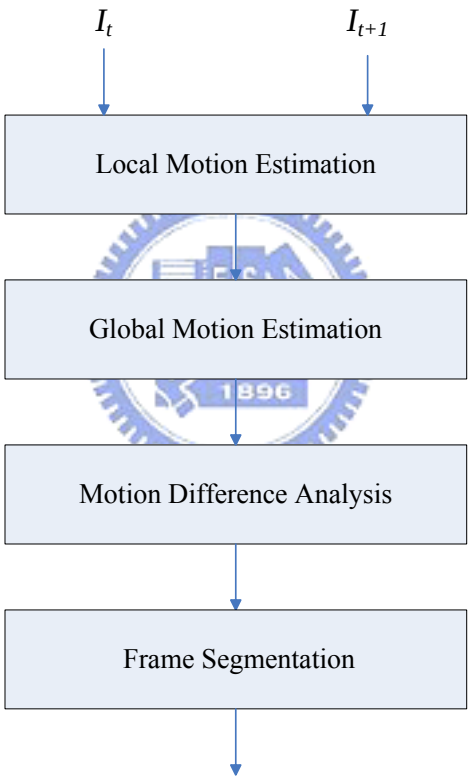
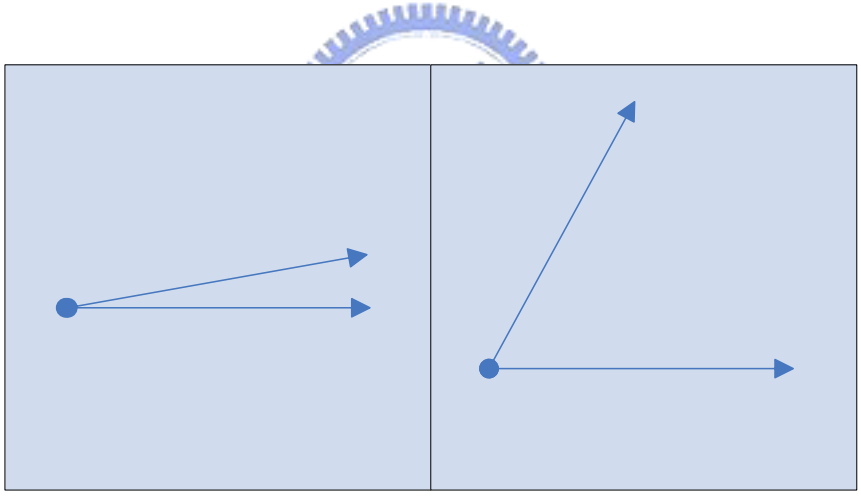


圖 24：視訊切割架構

圖 24 中視訊切割的第二步驟為全域運動之估測（Global Motion Estimation），所謂的全域運動是以攝影機的觀點做宏觀的計算，希望得到所有的物體在只有攝影機運動影響下的運動資訊（即把真實世界中所有物體皆當成靜止不動），在這部分會藉由攝影機運動參數來估測畫面中整體運動的情況。切割畫

面時，每個部分都要估測出其區域運動與全域運動，區域運動與全域運動的關係有兩種，一種為圖 25（a）所示，區域運動與全域運動差異較小，另外一種情況為圖 25 （b）所示，區域運動與全域運動差異較大。若區域運動與全域運動差異較小，則此部分為背景的機率較大，在真實世界中可能是靜止的物體；若區域運動與全域運動差異較大，則此部分可能為我們所定義的前景，因為物體不單只有攝影機運動，還有自身的運動。因此，為了進行視訊切割，必須要找出整張畫面中所有的區域運動及所有的全域運動才能做進一步的切割。在演算法的第三個步驟是要分析區域運動與全域運動的差異性，並訂定好評斷區域運動與全域運動之差異的臨界值，此臨界值會用來當作切割為背景或前景的標準。在演算法的最後一個步驟會根據前一步驟所訂定的標準來切割影像，若某一部份之區域運動與全域運動之差異大於所訂定的臨界值，則切割為前景，反之，切割為背景。



(a) 運動差異小　　(b) 運動差異大

圖 25：運動向量之差異性

5.2 區域運動估測與全域運動估測

在前一小節中介紹了視訊切割的概念與流程，本小節會針對前一小節所提到的區域運動估測與全域運動估測再做進一步的介紹。

5.2.1 區域運動估測使用區塊比對

在區域運動估測必須要找出兩張畫面中具有運動關係的對應點，考慮一般的視訊壓縮皆以“區塊比對為基礎”，為了視訊切割可以進一步的整合到視訊壓縮之中，本論文皆使用“區塊比對”當作“區域運動估測”之方法。區塊比對的一項優點就是較為簡單，但是也有其缺點，所列如下：

1. 遮蔽或消失：兩張影像中欲尋找對應關係的對應點，可能會因為遮蔽的現象而找不到真正有對應關係的點。另外若某物體原本在畫面裡，在下一個時刻卻移出了畫面，在這種情況下是無法找到正確的對應點。
2. 扭曲或變形：物體本身有嚴重的扭曲或變形，對應點也沒辦法找的很好。
3. 模稜兩可或物體邊緣：區塊比對是以灰階值的大小差異為判斷標準，找出灰階值差異最小的區塊。當區塊比對的區塊太小時，在很大的平順的區域中，由於灰階值差異太接近，可能會有對應點誤判的情形產生。另外當所使用的區塊太大時，在物體的邊緣常常會包含多個物體，因為每個物體的運動不一定一致，所以針對一個區塊只找一個對應點可能會有錯誤的情況產生。
4. 搜尋範圍：區塊比對時，搜尋的範圍不能無限制的擴張以免增加運算的複雜度；一般來說，搜尋的範圍在 16 個像素值最常被拿來使用，但在某些情況下實際物體的運動會超過 16 個像素，所以可能也無法找到最佳的對應點。

雖然區塊比對有上述的缺點，但是因為其演算法較為簡單，且視訊壓縮標準中也使用區塊比對，所以我們仍然使用區塊比對來估計區域運動。本論文所使用的區塊比對皆以 16×16 區塊為單位，而搜尋範圍均設定為±16 個像素。

5.2.2 全域運動估測使用 PTZ 攝影機運動模型

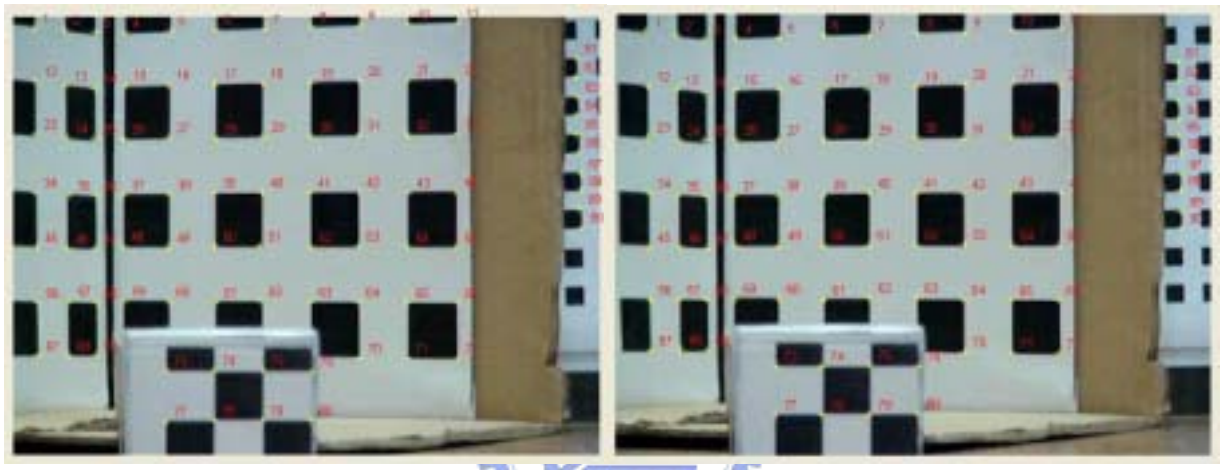
全域運動估測時，本論文使用了Peter J. Ramadge等人 [29][30]所提出的攝影機運動估測模型，此模型基本假設是攝影機運動只能有放大、縮小、上下或左右傾斜攝影，但是攝影機的中心必須固定於定點不可有平移運動，我們簡稱此模型為PTZ（Pan、Tilt、Zoom）運動參數模型，我們以此模型來近似一般攝影機運動之模型。

$$x' = \frac{p_1 x + p_2 y + p_3}{p_5 x + p_6 y + 1}$$
$$y' = \frac{-p_2 x + p_1 y + p_4}{p_5 x + p_6 y + 1} \qquad (5.1)$$

PTZ 運動估測模型如（5.1）式所示，其中（x , y）與（x', y'）分別為相鄰兩張畫面中具有運動對應關係的兩個點之影像座標，而 p_1、p_2、p_3、p_4、p_5 與 p_6 代表著運動模型的六個參數，此模型之推導可以在[29]中得到，值得注意的是座標（0,0）對應到的是影像畫面的中心。

在本小節，我們使用兩段符合 PTZ 運動的影片來驗證 PTZ 模型的準確度，為了估計攝影機的運動及其運動參數，每次共需要兩張畫面。在第一段影片中，物體均靜止不動，而攝影機則是單純的只有縮小的動作， 圖 26（a）與圖 26（b）分別為影片中攝影機運動前與攝影機運動後的兩張畫面，兩張畫面中各自有 90 個數字，代表著我們在兩張畫面中手動標示的 180 的點，以圖 26（a）中數字 1 的點為例，當攝影機經過運動後，圖 26（a）中數字 1 的位置會移動到圖 26（b）中數字 1 的位置，我們稱這樣的兩個點為一對“對應點”，其他數字也有同樣的對應關係，所以兩張畫面中共有 90 對具有對應關係的對應點，另外，為了稱呼方便，在圖 26（a）中的點皆稱為起點，在圖 26（b）中的點皆稱為終點。

為了估計 PTZ 模型參數，所使用的方法為“非線性回歸”（Nonlinear Regression)。將圖 26 中 90 對“對應點”隨機分為 60 對與 30 對兩群，其中 60 對的一群是做為估計攝影機參數使用，稱為訓練群（Training Set)，而 30 對的一群則是要驗證算出來的攝影機參數，稱為測試群（Testing Set)。把訓練群當作非線性回歸的輸入，經過非線性回歸的計算，會求出表 7 的六個相機參數，用此六個參數即可以描述圖 26 （a）與圖 26（b）之間的攝影機運動。



(a) 處理畫面 I_t　　(b) 參考畫面 I_{t+1}

圖 26：Zoom 影片中的手動標示點

假設圖 26 （a）中有一起點座標為（x ，y)，對應到圖 26 （b）終點座標（x', y'）；若將起點座標與表 7 的六個運動參數帶入（5.1）式，可求得新的終點座標（x'_n ，y'_n)。距離誤差（5.2）式可用來計算所求得的終點座標（x'_n ，y'_n）相對於手動標示座標（x', y'）之準確性，其誤差的單位為像素。表 8 統計出 60 對訓練群的終點誤差分佈情形，從表 8 可以看出經由攝影機參數所估得的終點誤差都小於兩個像素以下。表 9 統計出 30 對測試群的終點誤差分佈情形，可以看到誤差也在兩個像素以下。

$$err = \quad |x' - x'_n| + |y' - y'_n| \qquad (5.2)$$

表 7：圖 26 (a) 與圖 26 (b) 的攝影機參數

P_1	P_2	P_3	P_4	P_5	P_6
0.962542	-0.000416	-0.094951	0.176463	0	0.000026

表 8：訓練群終點誤差分佈（共 60 對）

	0<err≦0.5	0.5<err≦1.0	1.0<err≦1.5	1.5<err≦2.0	2<err	Total
Number of pairs	9	32	16	3	0	60

表 9：測試群終點誤差分佈（共 30 對）

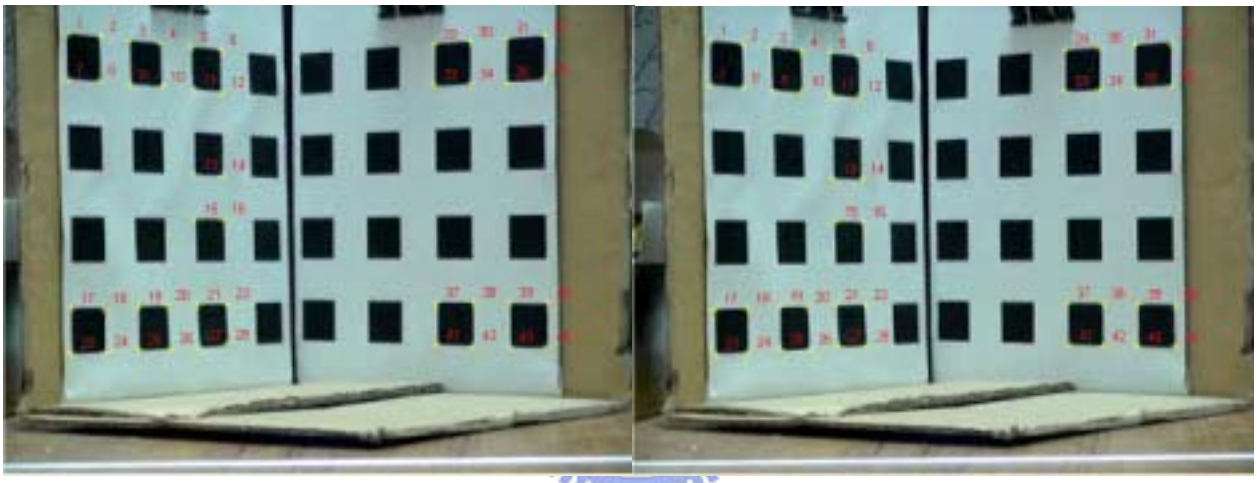
	0<err≦0.5	0.5<err≦1.0	1.0<err≦1.5	1.5<err≦2.0	2<err	Total
Number of pairs	8	10	8	4	0	30

估計攝影機的運動及其運動參數時，每次共需要兩張畫面及畫面中的對應點關係，為了進一步測試 PTZ 模型的誤差，我們總共估計了 5 組攝影機參數，此五組攝影機參數所造成的終點誤差之分佈統計於表 10 中，表中的 f0 f1 代表所估測的攝影機參數是畫面 0 與畫面 1 之間的參數，f1 f2 代表所估測的攝影機參數是畫面 1 與畫面 2 之間的參數，其他以此類推。表 10 中實驗結果顯示，終點誤差皆小於 2 個像素以下，這代表著當拍攝影片時的攝影機若約略符合 PTZ 運動模型，則計算出來的攝影機參數之精準度在兩個像素以內，另外誤差小於 1.5 像素的佔了全部的 94%，誤差小於 1 個像素的則佔了 70%。

表 10：終點之誤差統計（共 450 對）

	0<err≦0.5	0.5<err≦1.0	1.0<err≦1.5	1.5<err≦2.0	2<err	Total
f0 f1	17	42	24	7	0	90
f1 f2	20	41	22	7	0	90
f2 f3	27	40	21	2	0	90
f3 f4	23	40	20	7	0	90
f4 f5	20	45	19	6	0	90
	24%	46%	24%	6%	0%	

在我們拍攝的第二段影片中，物體一樣都靜止不動，而攝影機除了縮小的動作外還有往左傾斜攝影，圖 27（a）與圖 27（b）分別為影片中攝影機運動前與攝影機運動後的兩張畫面。在圖 27 的兩張影像中標示了 44 對“對應點”，將 44 對“對應點”隨機分為訓練群 24 對與測試群 20 對，而誤差函數同樣使用（5.2）式。訓練群與測試群的誤差結果統計在表 11 中，由表 11 可得知終點誤差也都小於兩個像素，而誤差小於 1.5 像素的佔了全部的 91%，誤差小於 1 個像素的則佔了 71%。



(a) 處理畫面 I_t　　(b) 參考畫面 I_{t+1}

圖 27：Pan、Tilt、Zoom 影片中的手動標示點

表 11：圖 27（a）與圖 27（b）所求得的終點誤差之統計(共 44 對)

	0<err≦0.5	0.5<err≦1.0	1.0<err≦1.5	1.5<err≦2.0	2<err
Training set	5	12	5	2	0
Testing set	6	8	4	2	0
	25%	46%	20%	9%	0%

圖 28 是 MPEG 組織所提供的測試影片 Mobile 的手動標示點，本章節的實驗會對圖 28（a）的畫面做切割。Mobile 影片的攝影機運動除了縮小以外，尚會往左平移，因此，Mobile 影片的攝影機運動並不完全符合 PTZ 攝影機運動模型，但是在此我們同樣用 PTZ 攝影機運動模型來近似其攝影機運動，並估計其

誤差。為了計算攝影機參數，分別在圖 28 兩張畫面中的背景部分手動標示了 35 對“對應點”，如圖 28（a）與圖 28（b）所示。將 35 對“對應點”隨機分為訓練群 25 對與測試群 10 對，訓練群所求得的攝影機參數如表 12 所示，並且利用測試群來驗證終點的誤差，誤差函數同樣使用（5.2）式。誤差的結果統計在表 13 中，由表 13 可得知終點誤差也都小於兩個像素，而誤差小於 1.5 像素的佔了全部的 94%，誤差小於 1 個像素的則佔了 71%。全域運動估測流程如圖 29 所示，而估計全域運動的方法總結如下：

1. 找到具有運動關係的對應點。
2. 輸入對應點並利用非線性回歸求得攝影機參數。
3. 將 I_t 上起點座標與攝影機參數帶入（5.1）式，求出全域運動在 I_{t+1} 上的終點座標。

(a) 處理畫面 I_t　　(b) 參考畫面 I_{t+1}

圖 28：Mobile 影片中的手動標示點

表 12：圖 28 (a) 與圖 28 (b) 的攝影機參數

P_1	P_2	P_3	P_4	P_5	P_6
0.986704	0.000443	2.848848	0.155756	0	0

表 13：圖 28 (a) 與圖 28 (b) 所求得的終點誤差之統計(共 35 對)

	0<err≦0.5	0.5<err≦1.0	1.0<err≦1.5	1.5<err≦2.0	2<err
Training set	5	12	6	2	0
Testing set	6	2	2	0	0
	31%	40%	23%	6%	0%

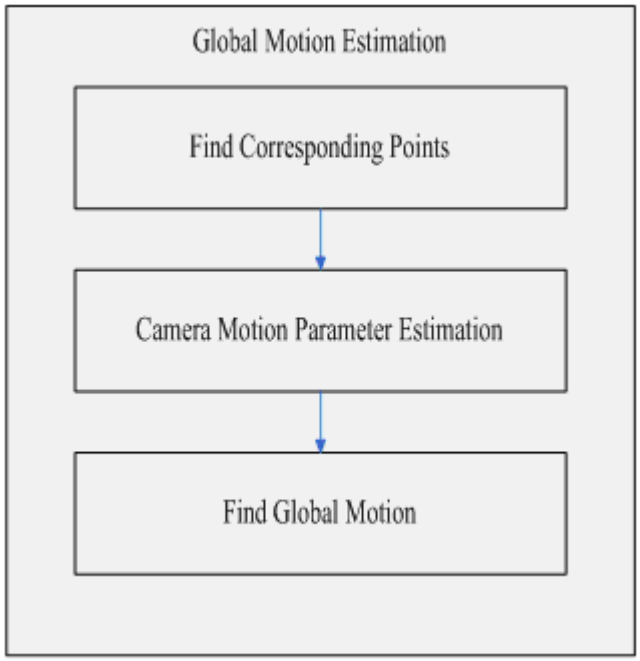


圖 29：全域運動估測流程圖

5.3 自動區域運動估測與手動全域運動估測

圖 30 為自動區域運動估測與手動全域運動估測之視訊切割架構圖，為了對圖 28(a)的 Mobile 畫面做切割，首先使用 16×16 大小的區塊比對方法求得圖 28（a）I_t 與圖 28（b）I_{t+1} 間的區域運動向量，因為 Mobile 為 325×288 的影像，所以一張影像共可切割成 396 個 16×16 的區塊，對於影像中的每一個區塊都必須使用區塊比對求得一個區域運動向量，所以對於一張畫面共有 396 的區域運動向

量。區域運動向量的起點為 I_t 畫面中現在處理區塊的左上角點座標，比對到 I_{t+1} 畫面中，區塊的左上角座標則為終點，以（x_l' , y_l' ）表示。圖 31（a）即為圖 28（a）參考圖 28（b），利用區塊比對後所找出的 396 個區域運動向量。

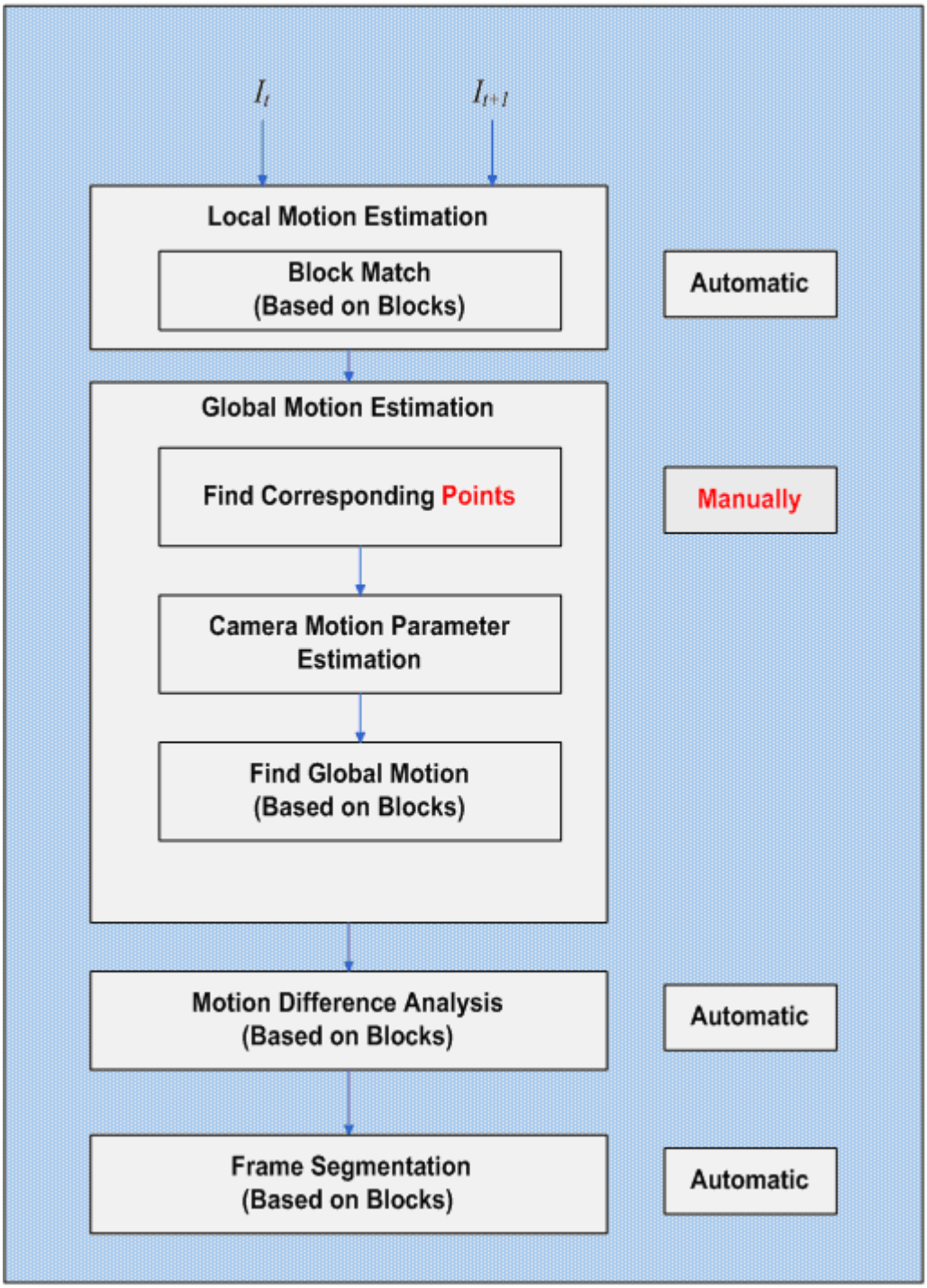
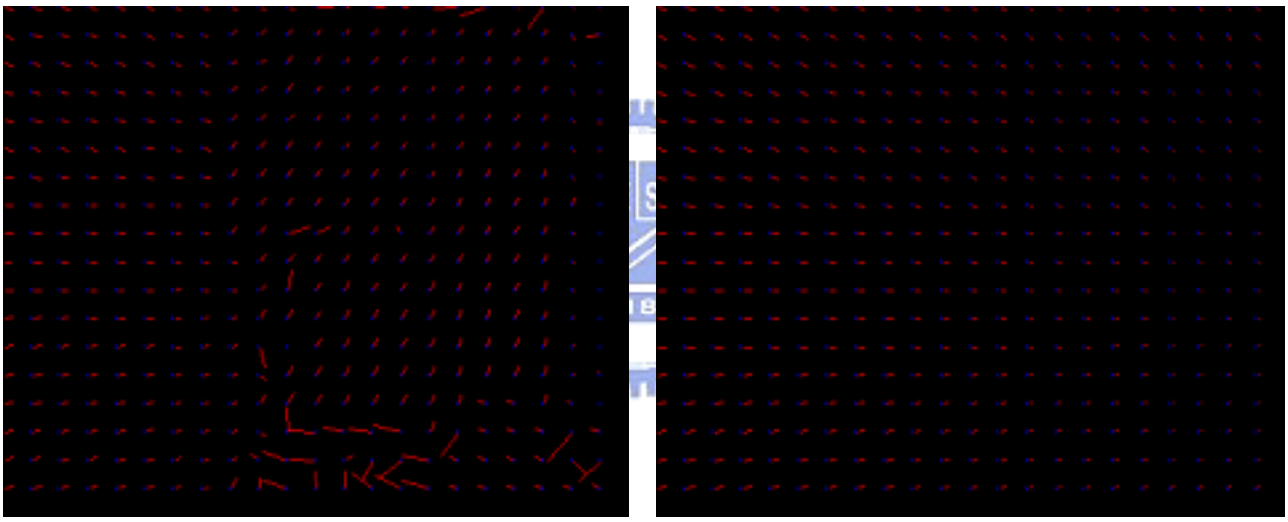


圖 30：自動區域運動估測與手動全域運動估測之視訊切割架構圖

PTZ 攝影機運動模型已經在 5.2 小節驗證過其準確性，為了攝影機參數之正確性，我們仍然使用手動標示的對應點所求出攝影機參數。此參數已經在 5.2 小節求出，列於表 12 中。由於區域參數之估測是以區塊大小為基礎，所以全域運動之估測亦以區塊大小為基礎，每一個區塊只需求得一個全域運動向量。在計算全域運動向量時，起點同樣為 I_t 畫面中現在處理區塊的左上角點座標，而全域運動向量之終點求法即是將起點座標帶入（5.1）式求出，I_{t+1} 畫面中上的全域運動終點座標以（x'_g, y'_g ）表示，圖 31（b）即為圖 28（a）參考圖 28（b），並經過全域運動向量估測所找出的 396 個區域運動向量。



(a) 區域運動向量　　(b) 全域運動向量

圖 31：區域運動向量與全域運動向量

圖 32 為以區塊為基礎的區域運動向量與全域運動向量之示意圖，每一個區塊皆會有一個全域運動向量與區域運動向量，兩個向量的起點皆為同一個，而終點則會依照其差異程度而有遠近的關係，定義兩個向量的差異程度使用（5.3）式估計：

$$d_{motion} = |x'_g - x'_l| + |y'_g - y'_l| \qquad (5.3)$$

其中（x'_g, y'_g ）代表全域運動終點座標，（x'_l, y'_l ）代表區域終點座標，d_{motion} 表

示區域運動向量與全域向量的差異程度，單位為像素。當 d_{motion} 越大時，物體不同於攝影機運動的機率越高，也就是說實際上物體本身是在運動的，此時就可被劃分為前景；當 d_{motion} 越小，物體相似攝影機運動的機率越高，此時真正的物體是靜止不動的，物體看起來會運動的原因是因為攝影機的運動。由於區域運動估計是以區塊為單位，所以後續的切割均是以區塊為單位。

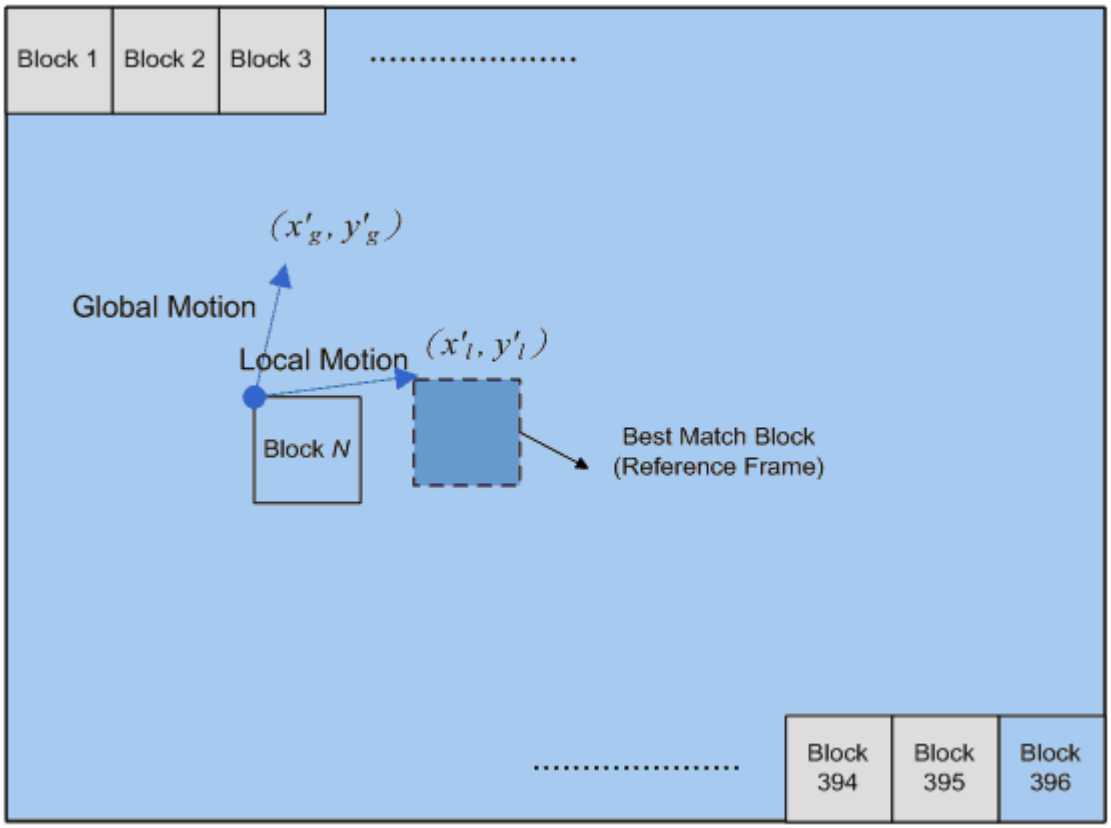


圖 32：16×16 區塊之區域運動向量與全域運動向量

對於圖 28（a）所求出的 396 個 d_{motion} 值如圖 33 所示，縱座標為區塊總個數，每個橫軸 d_{motion} 值所統計的是具有相同 d_{motion} 值的區塊個數，從圖 33 可以發現大多數的區塊集中在 d_{motion} 為 1 與 d_{motion} 為 5 附近，這表示畫面中有一群區塊的區域運動向量與全域運動向量差異很小，而另外一群則差異很大。在切割影像時，只要訂定一個 d_{motion} 的臨界值即可對影像作切割，當 d_{motion} 小於臨界值即判斷為背景，d_{motion} 大於臨界值則判斷為前景。圖 34 為使用自動區域運動估測

與手動全域運動估測之切割結果，圖 34（a）為取臨界值 1.4，圖 34（b）為取臨界值 3.4，圖中紅色方塊所遮罩的地方是被判斷為前景的區塊；在 Mobile 影片中會自行運動的物體即月曆、球與火車，從圖中的結果可以看到有相當不錯的初步切割，然而在前景和背景部分各有一些被誤判的地方，這是因為區域運動估計之偏差所造成，在後續的章節我們會將雜訊去除而使得切割更加完整。圖 34（a）與圖 34（b）雖然只有顯示臨界值 1.4 與 3.4 的結果，但是其實介於 1.4 與 3.4 間的臨界值之切割結果亦相當不錯。

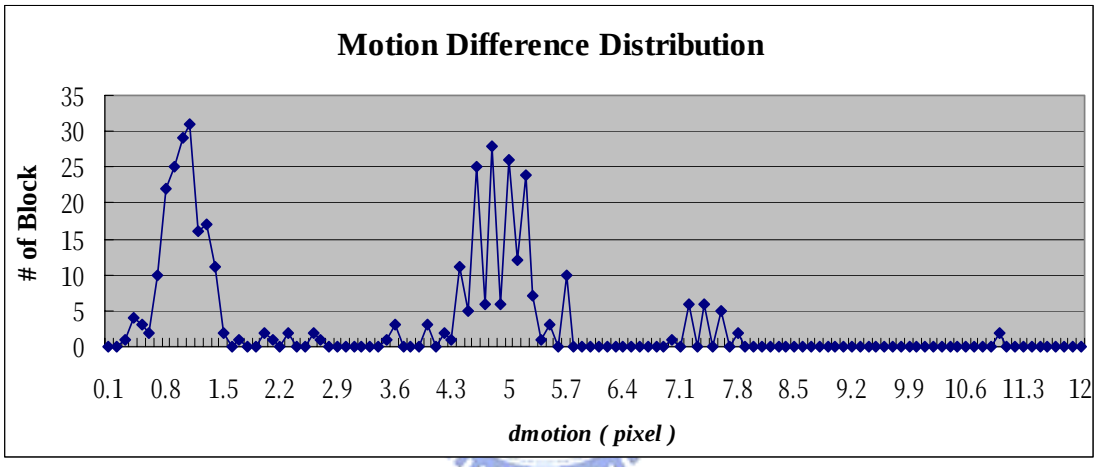


圖 33：Mobile 畫面之自動區域運動估測與手動全域運動估測 d_{motion} 值

(a) threshold 1.4　　(b) threshold 3.4

圖 34：Mobile 畫面切割結果之一

5.4 自動區域運動估測與半自動全域運動估測

為了達到自動化的目的，更改圖 30 之視訊切割架構如圖 35 所示，其中的差異在於圖 30 的全域運動估測之輸入對應點是由手動標示，而圖 35 引用“區域運動估測”所產生的對應點關係（更正確的說，是“對應區塊”的左上角座標）當成“全域運動估測”之輸入，如此一來可省掉手動標示的麻煩與時間。圖 35 之視訊切割架構的第一步同樣是使用 16×16 區塊大小的區塊比對方法，進行“全域運動估測”時，將“區域運動估測”所產生的對應關係當成全域運動估測的輸入。因為在“區域運動估測”段尚未做切割，所以無法區分前景與背景，若計算攝影機參數混入太多的前景對應點時，則估測出的攝影機參數會有嚴重的偏差，因此我們雖然自動產生了對應點關係，於是在估測全域運動的第一階段：Find Corresponding Blocks 中，我們必須手動圈選屬於背景部分的對應點（或稱為對應區塊的左上角座標點），做為計算攝影機參數之使用，針對這一部份，在後續的章節我們提出改進的方法。

同樣的，我們以圖 28（a）與圖 28（b）做為影像切割演算法的輸入，然後針對圖 28（a）的畫面做切割。第一個步驟同樣使用 16×16 區塊大小的區塊比對方法先求得區域運動向量，結果如圖 36（a）所示，此結果與圖 31（a）相同，因為使用的是同一個方法，並且針對同樣的輸入影像作處理。第二個步驟用人工的方式選出背景部分（月曆、球、火車以外的區域）的對應點並計算出攝影機參數，攝影機參數如表 14 所示，計算出攝影機參數之後將 396 個區塊的起點座標分別帶入（5.1）式求出全域運動向量之終點。全域運動向量如圖 36（b）所示，雖然圖 36（b）與圖 31（b）是使用不同方法求出，但是相當相似，以人眼亦分辨不出其差異。此時，已經求出區域運動向量與全域運動向量，接下來便對運動向量的差異程度做分析並切割影像。同樣的，使用（5.4）式來估計區域運動向量與全域運動向量之差異程度，圖 37 統計了運動向量的差異程度分佈，圖中亦

分成了兩個集團，因此對於切割相當容易。使用人眼觀察之較好切割結果介於臨界值 1.5 與 3.2 之間，在圖 38 中我們只列出臨界值 1.5 與臨界值 3.2 的結果。

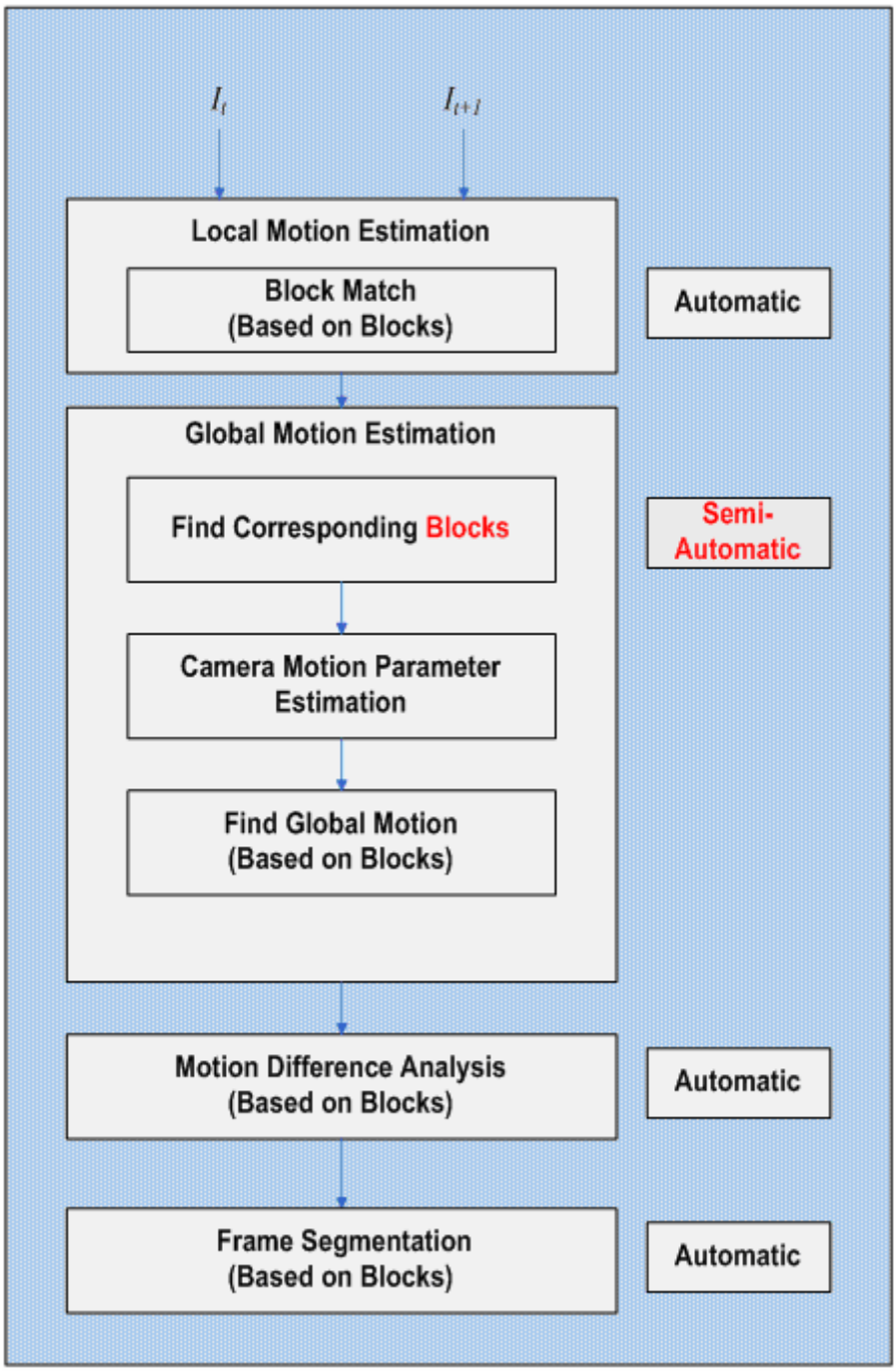
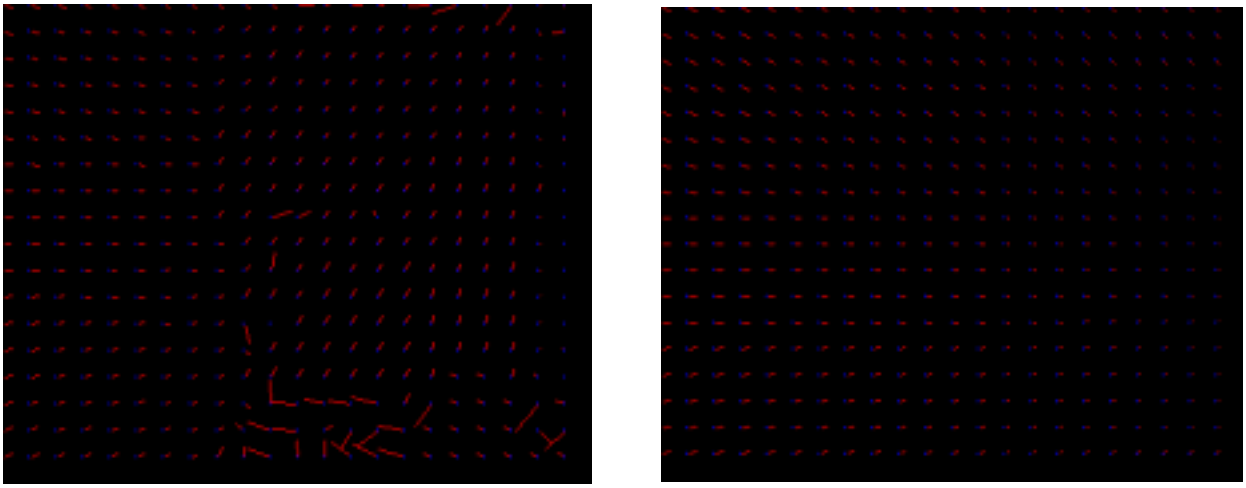


圖 35：自動區域運動估測與半自動全域運動估測之視訊切割架構圖

表 14：半自動全域運動估測計算出的攝影機參數

P_1	P_2	P_3	P_4	P_5	P_6
0.989884	-0.000119	2.476433	-0.133236	-0.000018	-0.000001



(a) (b)

圖 36：自動區域運動向量與半自動全域運動向量

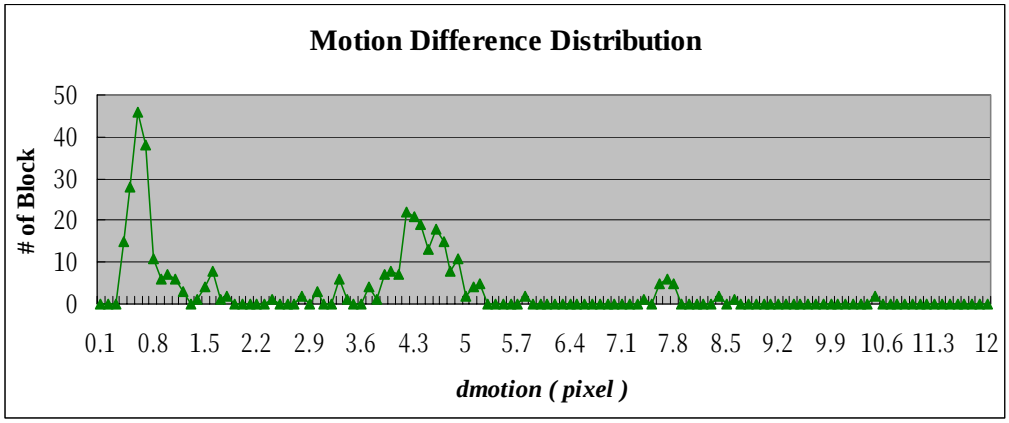


圖 37：Mobile 畫面之自動區域運動估測與半自全域運動估測 d_{motion} 值

(a) threshold 1.5 (b) threshold 3.2

圖 38：Mobile 畫面切割結果之二

5.5 自動區域運動估測與自動全域運動估測

為了達到自動化的目的，我們進一步改善 5.4 小節的視訊切割架構。新的視訊切割架構如圖 39 所示，不同的地方在於不必再使用人工的方式選出背景的區域。新的架構中，“全域運動估測”之輸入是“區域運動估測”之所有輸出，此輸出不必區分前景背景。然而由於估計攝影機參數時，混入了許多的前景的對應區塊（或對應區塊的左上角座標點），所以計算出的參數會有偏差，必須進一步更新。全域運動估測時，偏差的參數修正流程如圖 40 所示，茲說明如下。

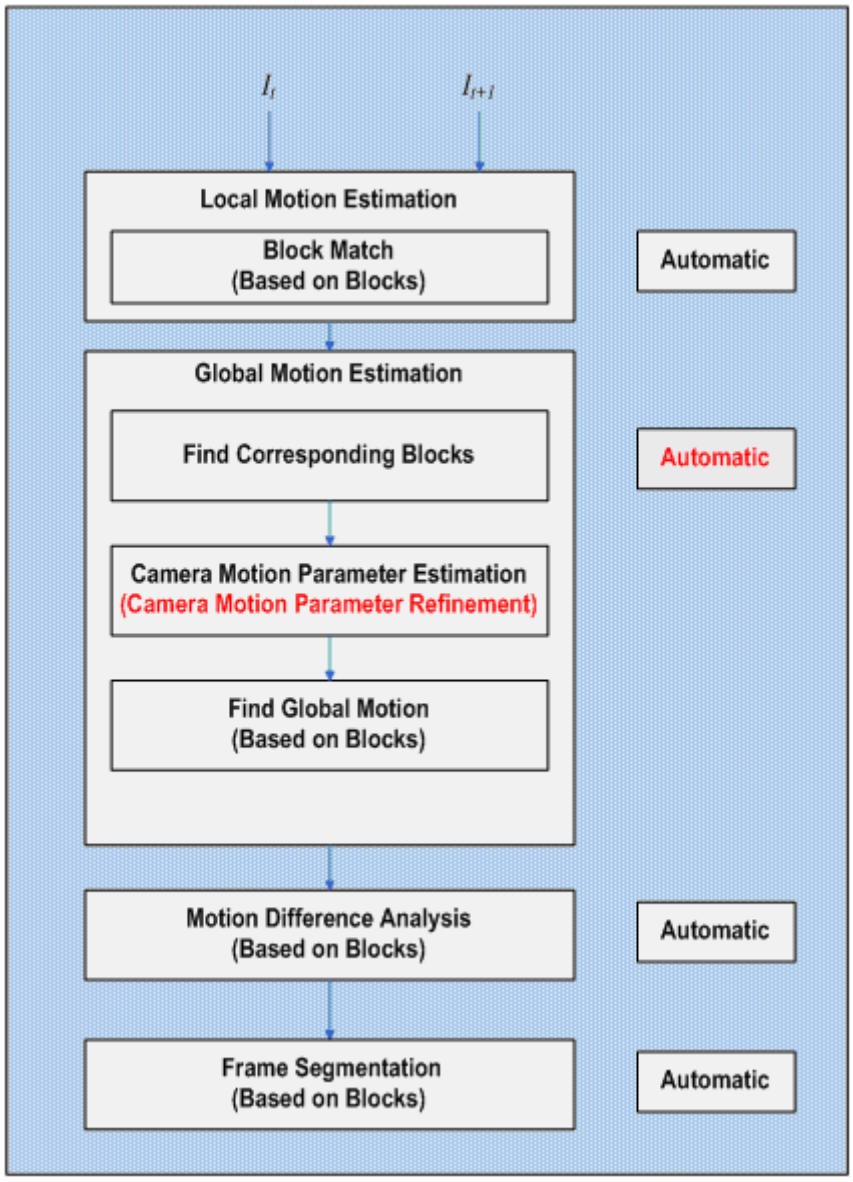


圖 39：自動視訊切割架構圖

通常來說，背景的範圍會佔了大半數的畫面，或者在好幾群物體的運動當中，背景通常是最大的一群。基於這個假設下，可以達到偏差攝影機參數之修正。因為背景佔了大半數的畫面，因此在計算攝影機參數時，此一參數之估測雖然不完全正確，但是會造成背景包含 d_{motion} 小的區塊會比較多，前景包含 d_{motion} 小的區塊會比較少。我們將 d_{motion} 較小的區塊取一定的數量（35%）再重新記算一次攝影機參數，此時 d_{motion} 較小（35%）的區塊中，背景佔的比例會比原先的大，前景干擾也變少了，所以此次算出來的 PTZ 參數會更接近真正的背景，如此一直重複同樣的步驟至收斂便可以求得真正的背景參數，達到偏差攝影機參數之修正。

針對圖 28（a）的畫面做切割，圖 41 為每個回合修正偏差參數所使用的對應點（對應區塊的左上角座標點），表 15 為各個階段修正後的攝影機參數。首先，在第一回和中，用整張畫面的 396 個區塊之對應關係求出表 15 中 loop 0 欄位的攝影機參數，由於一開始攝影機參數一定不會收斂，所以繼續執行下一個步驟。在下一個步驟中，利用已求出的參數來計算出臨時的全域運動向量與 396 個區塊的 d_{motion} 值，其中 d_{motion} 較小的 35% 區塊為圖 41（a）中所標示的點（對應區塊的左上角座標），在本回合的結束，挑選圖 41（a）中標示的對應區塊當作下一回合所要使用的資料。第二回合的開始，用圖 41（a）中的對應區塊重新計算攝影機參數，算出的參數列在表 15 中 loop 1 欄位，並再利用此參數估計新的全域運動向量與 d_{motion} 值，之後重新挑選擁有較小 d_{motion} 值的對應區塊如圖 41（b）所示，就可再重新計算與更新攝影機參數。一直重複上述步驟直到攝影機參數收斂可求得最後的更新參數，如表 15 所示，在 loop 19 便可以發現攝影機參數已經收斂。在修正偏差參數時，第一回合所挑的點為圖 41（a），可以發現除了右下角的點是屬於火車的部分點，其他大部分的點都是背景點；由圖 41 (a) 至圖 41（f）中，挑到火車的點也逐漸變少了，這表示參數正逐漸向正確的方向更新，最後收斂結果所使用的對應區塊如圖 41（f）所示，可以發現此時計算攝影機參數時所用的區塊皆屬於背景區塊。在修正偏差參數的過程中，雖然要到

loop 19 才會完全收斂，但是從表 15 中可以發現，在 loop 8 之後，攝影機運動參數的變動已經相當小，為了節省程式計算的時間，亦可以考慮使用 loop 8 之後的攝影機運動參數來計算全域運動向量。

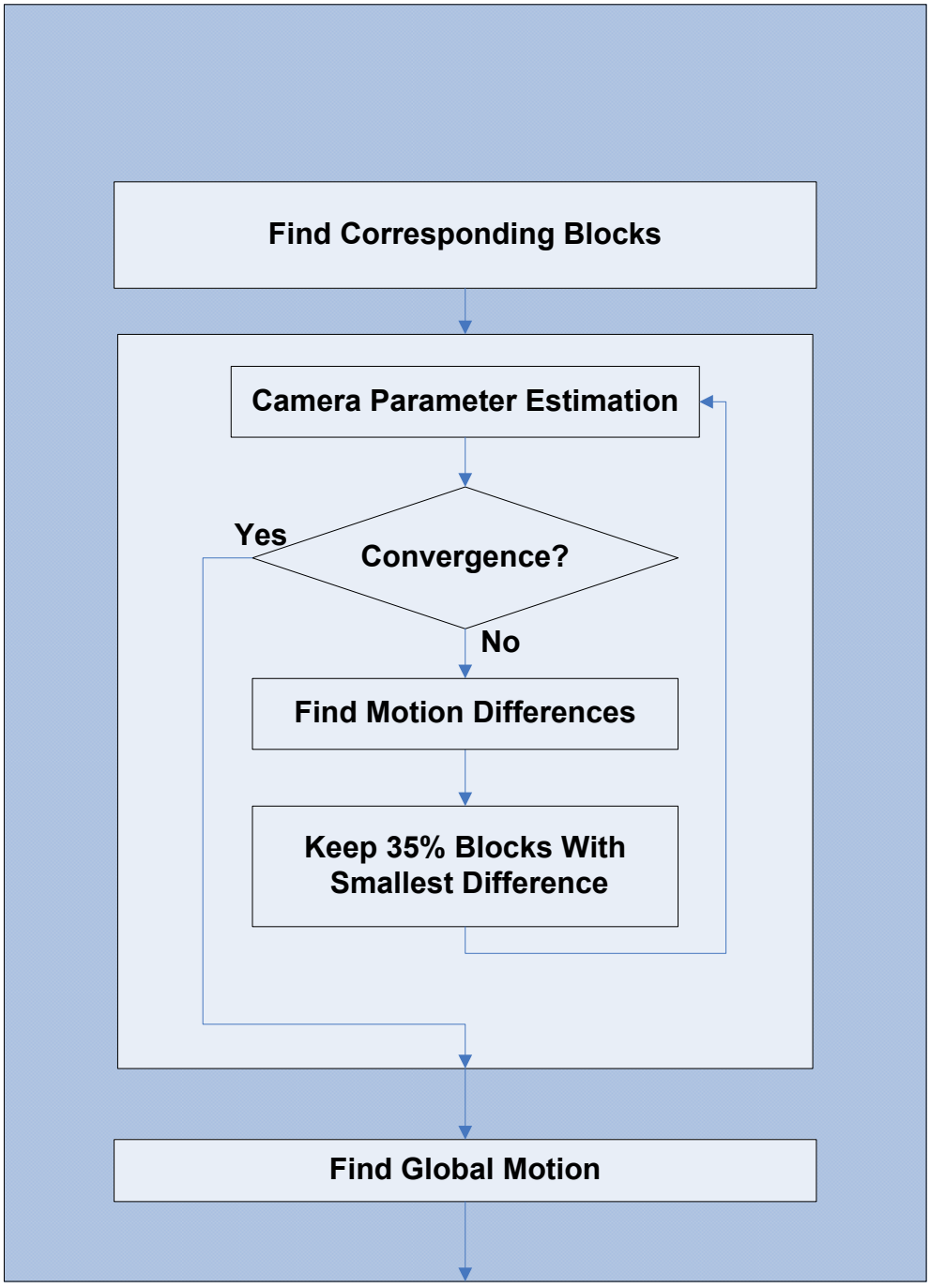
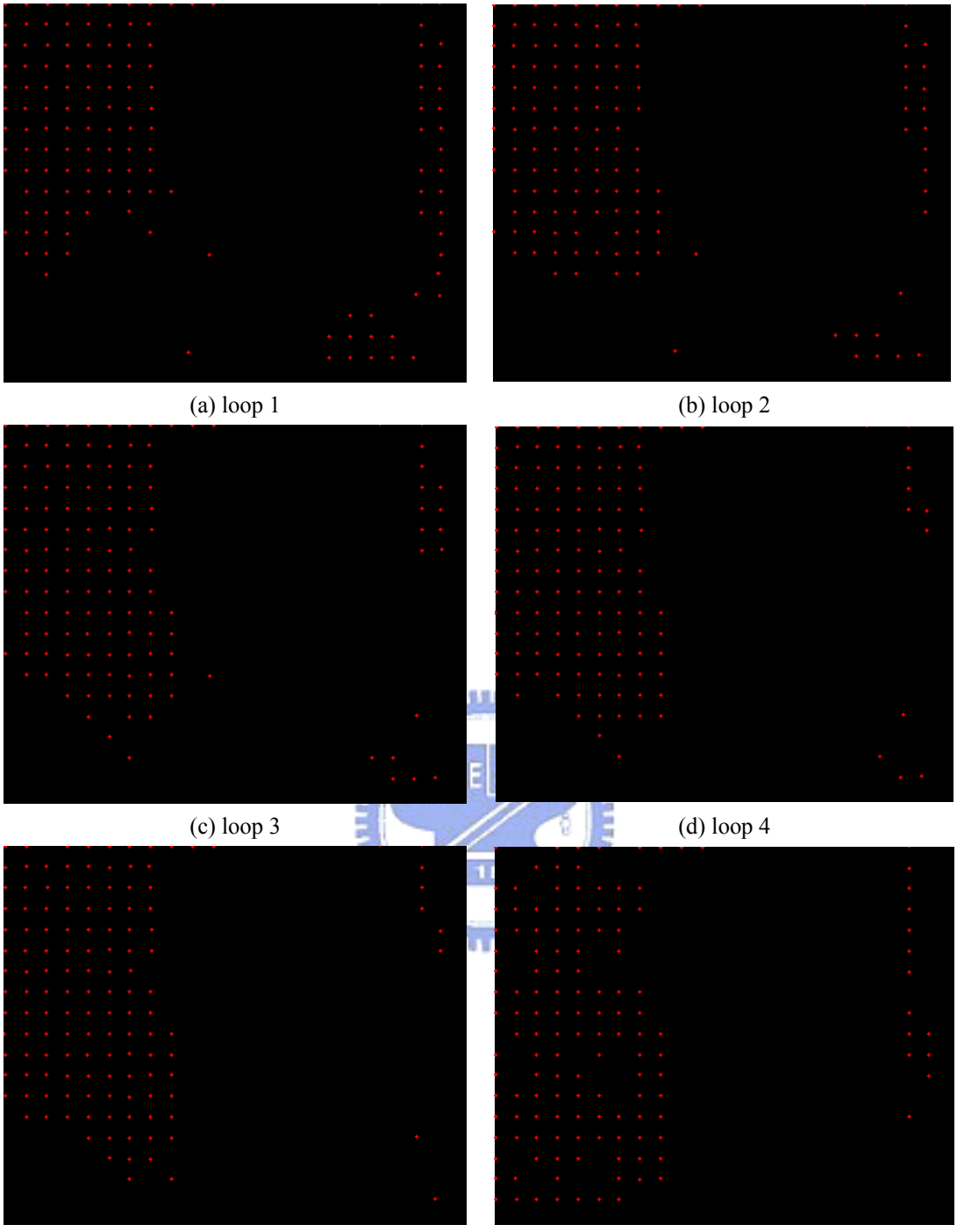


圖 40：全域運動估計與偏差參數修正流程圖



(a) loop 1　(b) loop 2

(c) loop 3　(d) loop 4

(e) loop 5　(f) loop 19

圖 41：修正偏差參數所使用的對應點

表 15：修正的偏差參數

	P_1	P_2	P_3	P_4	P_5	P_6
loop 0	0.980559	-0.01505	-1.68773	-1.20126	-0.000126	-0.000049
loop 1	0.979624	-0.00935	-0.58364	-0.5998	-0.000101	0.000041
loop 2	0.978963	-0.00629	0.337311	-0.00797	-0.000065	0.000083
loop 3	0.978417	-0.00585	0.811989	0.290868	-0.000044	0.000098
loop 4	0.978807	-0.00551	1.161938	0.478087	-0.000025	0.000092
loop 5	0.979354	-0.00506	1.438223	0.504008	-0.000011	0.000083
loop 6	0.980787	-0.00333	1.768993	0.316352	0	0.000065
loop 7	0.983995	-0.00078	2.203469	-0.04101	0.000007	0.000025
loop 8	0.985246	-0.00081	2.348296	-0.01683	0.000008	0.00001
loop 9	0.985654	-0.00102	2.338605	0.009382	0.000005	0.000011
loop 10	0.985893	-0.0011	2.329482	0.023306	0.000002	0.000011
loop 11	0.986078	-0.00119	2.343406	0.03758	0.000002	0.000011
loop 12	0.986811	-0.00089	2.366261	-0.00891	0.000002	0.000008
loop 13	0.987109	-0.00097	2.372708	-0.00567	-0.000002	0.000008
loop 14	0.987211	-0.00097	2.362678	-0.00726	-0.000005	0.000008
loop 15	0.987334	-0.00108	2.36139	0.01172	-0.000005	0.000008
loop 16	0.987467	-0.00098	2.357213	0.000915	-0.000006	0.000008
loop 17	0.987516	-0.00088	2.365738	-0.01772	-0.000006	0.000007
loop 18	0.987549	-0.00081	2.372579	-0.03008	-0.000006	0.000006
loop 19	0.987549	-0.00081	2.372579	-0.03008	-0.000006	0.000006
loop 20	0.987549	-0.00081	2.372579	-0.03008	-0.000006	0.000006
loop 21	0.987549	-0.00081	2.372579	-0.03008	-0.000006	0.000006
loop 22	0.987549	-0.00081	2.372579	-0.03008	-0.000006	0.000006

最後，利用表 15 中 loop 19 的攝影機運動參數可求全域運動向量，並得圖 28（a）的 396 個 d_{motion} 值，其分佈如圖 42 所示。我們很容易的可以區分出前景與背景的區塊，切割的結果如圖 43（a）與圖 43（b）所示。而圖 43（c）與圖 43（d）是使用表 15 中 loop 10 的攝影機運動參數求得的全域運動向量，並對 d_{motion} 取不同的臨界值所產生的切割結果，可以看到結果亦相當不錯。圖 44 為另一段影片中的兩張畫面，已知影片中手持著鋁箔包運動的部分為前景，其他的部分則為背景，圖 45 為此影片使用本小節自動切割方法的結果，圖 45（a）列出臨界

值 1.7 的結果，圖 45（b）列出臨界值 3.5 的結果，d_{motion} 值則統計於圖 46 中，從圖 46 可以看出背景部分有很好的集中效果，而前景部分因為所佔區域較小，所以沒有累積相當多的區塊，但是前景部分之運動仍與攝影機運動差異很大，所以 d_{motion} 值亦很大，因此很容易切割。由於影片中平滑與規則的區域較大，所以區塊比對的時候誤差較多，因為即使這些區域是屬於運動的物體，但是進行“區塊比對”估計物體運動時，由於區域特徵上的不足，運動向量容易被誤判成 0，這造成部分“區域運動向量”估計上的不精準，最後也影響了切割的結果。

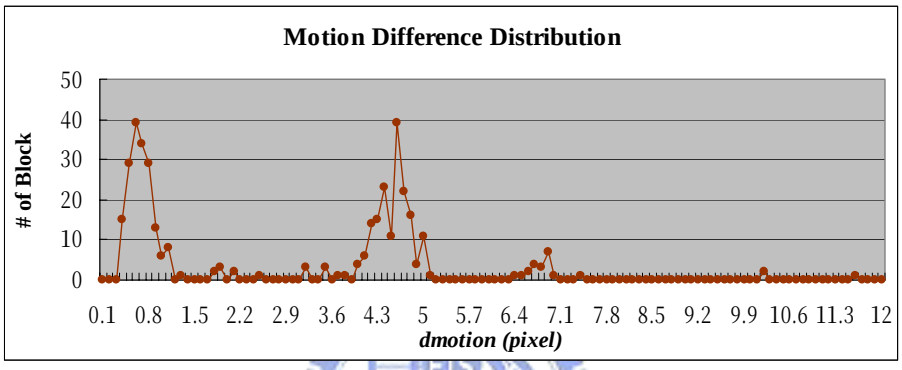


圖 42：Mobile 畫面之自動區域運動估測與自全域運動估測 d_{motion} 值

(a) loop19, threshold 1.5

(b) loop19, threshold 3.2

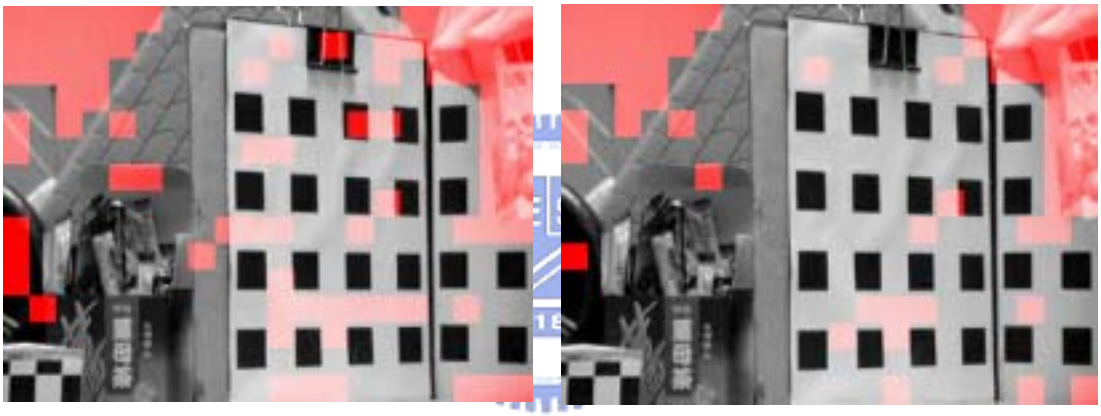
(c) loop10, threshold 1.5

(d) loop10, threshold 3.2

圖 43：Mobile 畫面切割結果之三

(a) 處理畫面 I_t　　(b) 參考畫面 I_{t+1}

圖 44：欲切割的 PTZ 影片



(a) loop10,threshold 1.7　　(b) loop10, threshold 3.5

圖 45：PTZ 影片切割結果

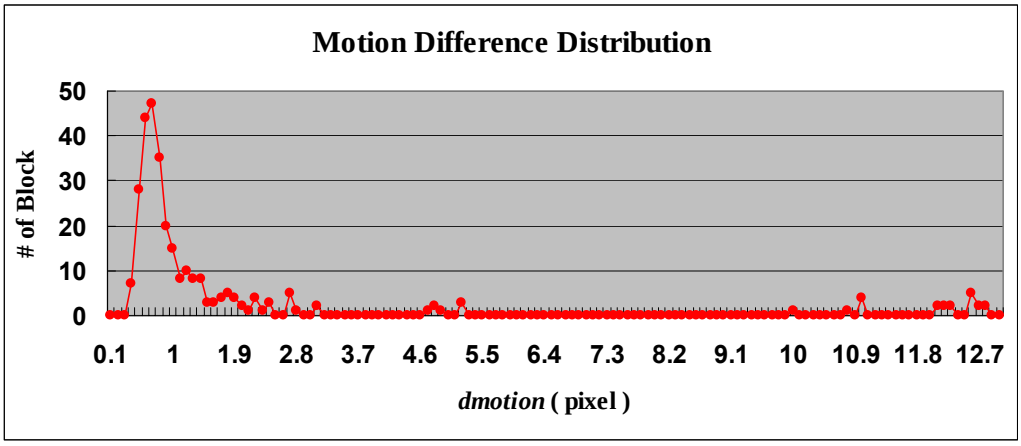


圖 46：圖 44 使用全自動切割方法之 d_{motion} 值

5.6 結合小區域雜訊之去除之自動視訊切割方法

在本小節使用了一個簡單方法去除雜訊。對於初步已經切割好的影像，如圖 43 (a)，重新檢視 396 的區塊被判斷為前景或背景的情況，對於現在處理的區塊，若周圍的區塊大多為前景則將本身區塊更新為前景，反之，則更新為背景，當周圍前景區塊與背景區塊個數相同，則不更新。要注意的是，被更新的區塊不應該影響下一個處理區塊之更新，在檢視周圍前景區塊或背景區塊之多寡時，是引用更新前的切割的成果。對圖 43（a）去除雜訊之切割結果如圖 47（a）所示，圖 45（b）去除雜訊之切割結果如圖 47（b）所示，其中較大的雜訊是無法去除的。

(a)：圖 43（a）去除雜訊　　(b) ：圖 45（b）去除雜訊

圖 47：去除雜訊之切割

5.7 不同參數設定的影響與實驗結果

在切割視訊影像時，d_{motion} 之臨界值的選取會影響到切割的結果，為了達到自動視訊切割之目的，必須要訂定一個合適的臨界值。另外，在想要切割的影像中，雖然存在著不同運動的物體，但是物體的運動卻與攝影機運動非常相似，造成求出的 d_{motion} 差異不大，此時也會切割的不好，在本章節將對這些影響的因素列入考慮，並分析實驗的結果。

5.7.1 運動向量差異 d_{motion} 之臨界值選取

在切割的影像時， 因為前景區域與背景區域在運動上有本質的差異，造成前景部分的運動向量差異（d_{motion}）會較大，從前幾個小節的 d_{motion} 分佈圖可以發現，前景區塊與背景區塊有各自的群聚效應，且兩個集團的 d_{motion} 值有一定的差異，造成在取臨界值時，有一段範圍的切割結果都相當不錯，所以很容易切割。另外在 5.2.2 小節中，我們用了三段不同的影片來評估 PTZ 模型之誤差，從實驗結果也可以發現誤差多在 2 個像素以下，所以在自動切割時取臨界值為 2 是一個可行的設定。

5.7.2 參考畫面 I_{t+1} 的選取與切割結果之關係

每次進行切割時，共需要輸入兩個畫面，分別為"所要切割的畫面" I_t 與其"參考畫面" I_{t+1}，圖 48 為 Mobile 影片中的六張影像，假設圖 48（a）為我們現在所要切割的影像 I_t，而圖 48（b）至圖 48（f）可任選一張畫面做為我們參考的影像 I_{t+1}。當然，亦可以選取其他的畫面當作參考影像，但是參考影像的選擇可能影響到切割結果的好壞。

在前面的章節，為了簡化問題，我們不針對參考畫面的選擇做討論。從 5.2 至 5.5 小節切割 Mobile 檔案時，均以圖 28（a）當作要切割的畫面 I_t，而圖 28（b）則當成參考畫面 I_{t+1}。其實圖 28（a）即為圖 48（a），是 Mobile 檔案中的第 0 張畫面，圖 28（b）則與圖 48（f）相同，為 Mobile 檔案中的第 6 張畫面。為了瞭解不同參考畫面對切割結果的影響，圖 49 列出了利用不同參考畫面所產生出來的區域運動向量與全域運動向量，其中圖 49 所顯示的全域運動向量是用本論文在 5.5 小節所提出的方法，並取 loop10 的攝影機運動參數求得的結果，由於所求得的全域運動向量並非整數，因此圖 49 的全域運動向量只顯示四捨五入後的結果。從圖 49 可以發現參考畫面越靠近，各個物體的運動向量越小；另外不管參考畫面的遠近，背景部分的區域向量與全域向量之差異 d_{motion} 大多在兩個像素以下（因為全域向量是利用背景部分的區域向量所求出，並且攝影機運動模型的誤差在兩個像素以下）；而前景的部分在參考畫面越遠離時，其區域向量與全域向量的差異 d_{motion} 較大，但是當參考畫面越近，由於某些物體的運動量會變小，造成了這些物體的 d_{motion} 也較小，若前景部分的 d_{motion} 值小到兩個像素以下，與背景部分混和在一起，此時就無法完整的從畫面中切割出前景。

如圖 50 所示，列出了使用不同參考畫面所求得的 d_{motion} 分佈，在圖 50（a）與圖 50（b）的 d_{motion} 分佈，前景與背景並沒有分成兩個集團，其切割結果如圖 51（a）與圖 51（b）所顯示，並不是很理想。而圖 50（c）的 d_{motion} 分佈已逐漸分成兩個集團，但是前景的集團並沒有完全大於 2 個像素，因此切割結果如圖 51（c）所示，仍有部分切割錯誤。最後，在圖 50（d）到圖 50（f）中的前景集團開始逐漸遠離背景集團，因此可以很容易的切割出前景與背景，切割的結果顯示在圖 51（d）到圖 51（f），由圖中可以看到切割的結果相當不錯。

(a) Frame 0

(b) Frame 1

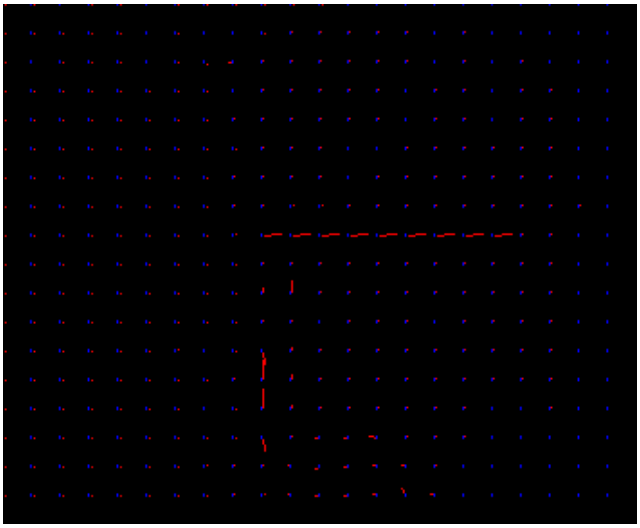
(c) Frame 2

(d) Frame 3

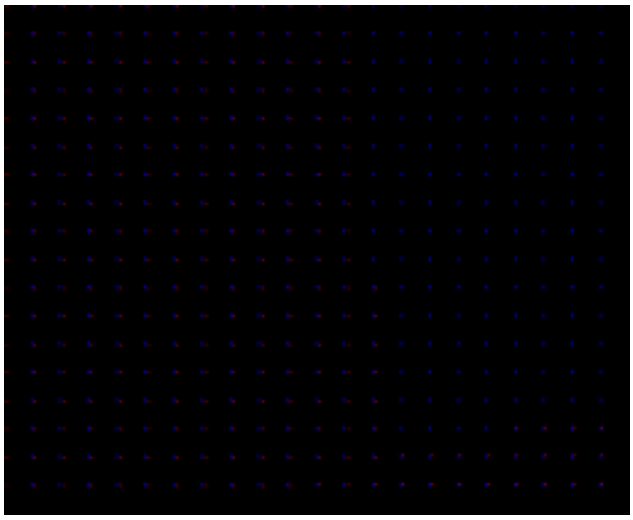
(e) Frame 4

(f) Frame 6

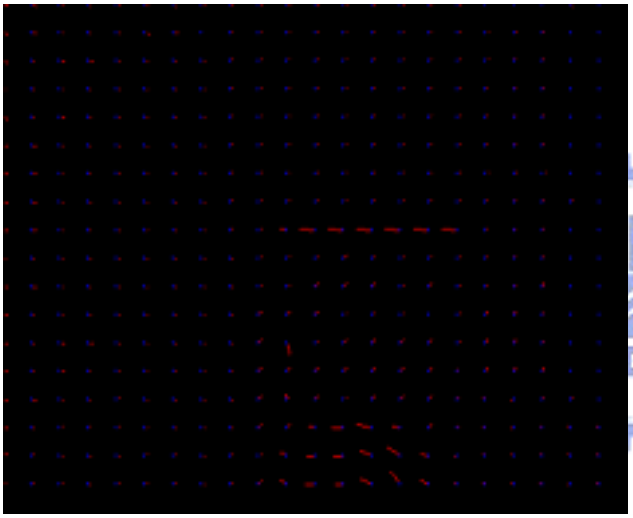
圖 48：Mobile 影片中的六張影像



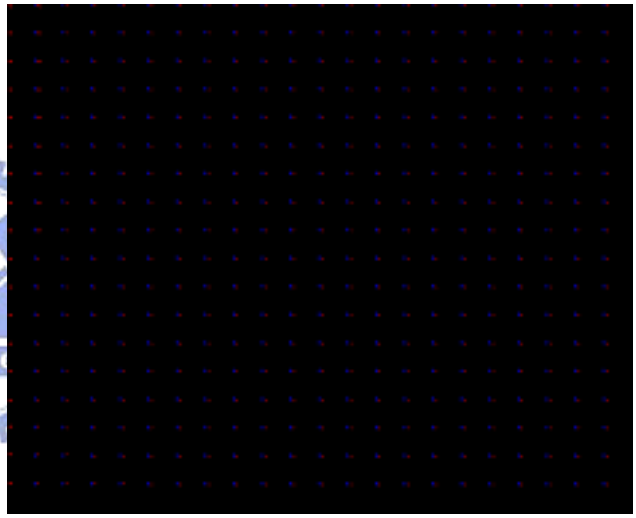
(a) 參考 Frame 1 所求得的區域運動向量



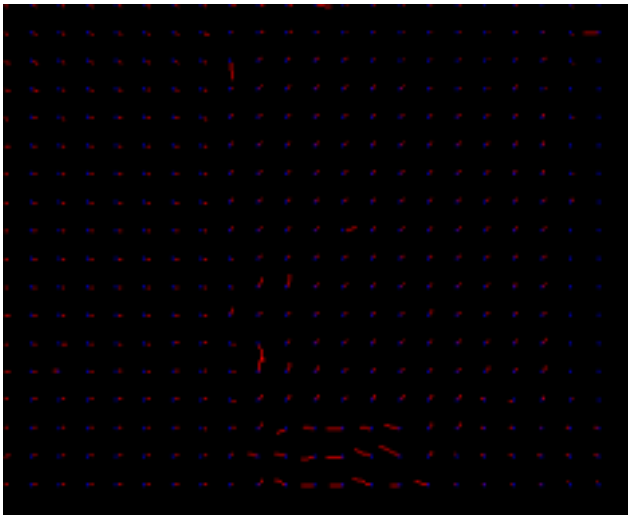
(b) 參考 Frame 1 所求得的全域運動向量



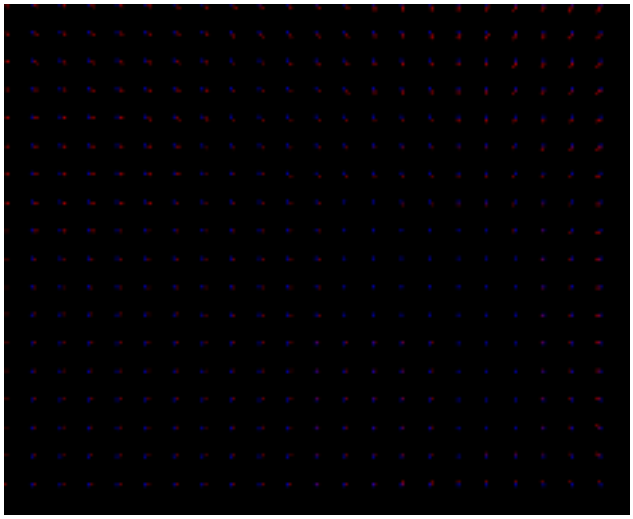
(c) 參考 Frame 2 所求得的區域運動向量



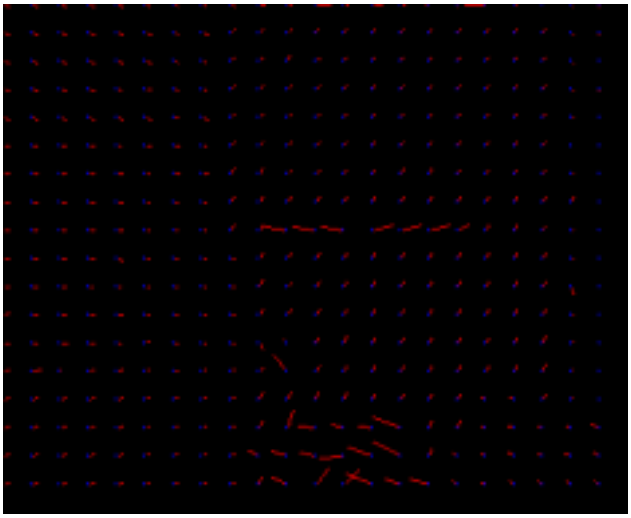
(d) 參考 Frame 2 所求得的全域運動向量



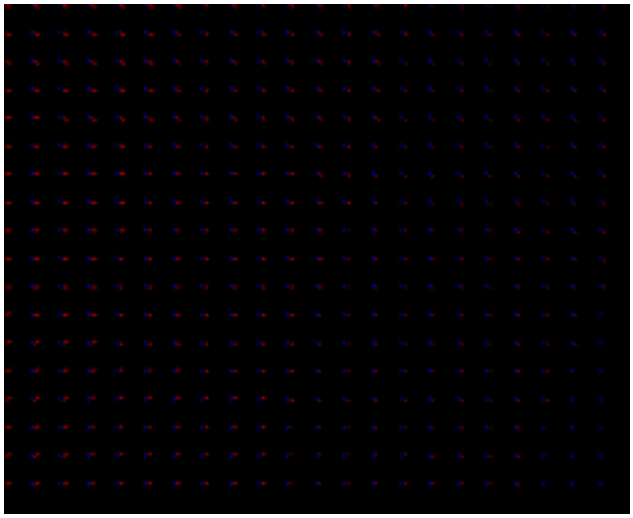
(e) 參考 Frame 3 所求得的區域運動向量



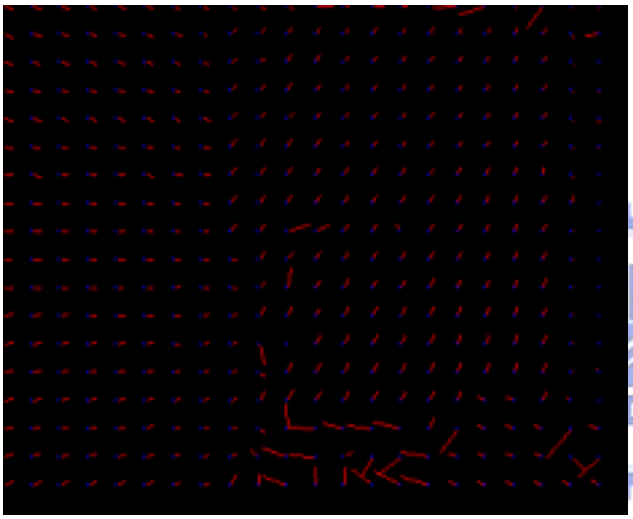
(f) 參考 Frame 3 所求得的全域運動向量



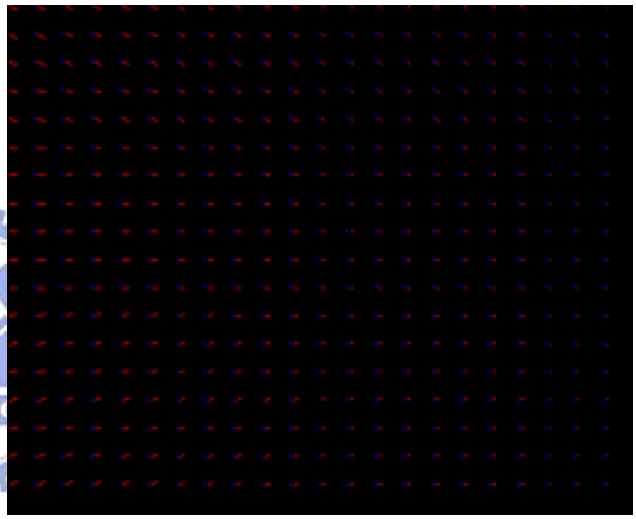
(g) 參考 Frame 4 所求得的區域運動向量



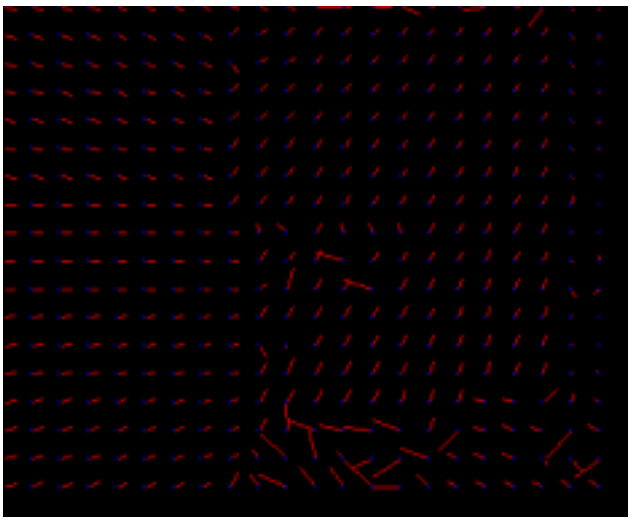
(h) 參考 Frame 4 所求得的全域運動向量



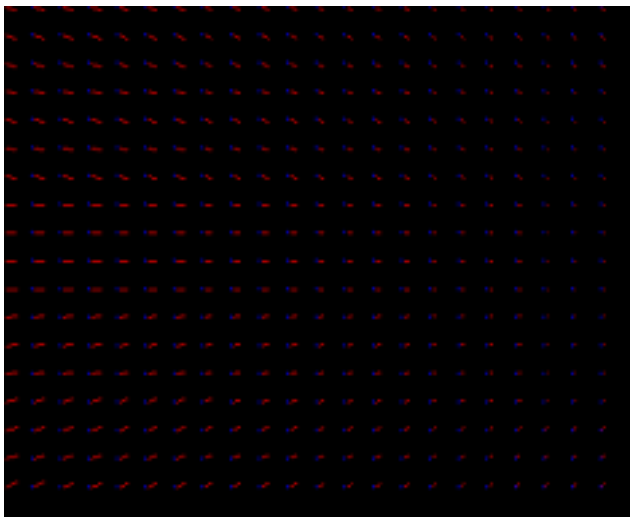
(i) 參考 Frame 6 所求得的區域運動向量



(j) 參考 Frame 6 所求得的全域運動向量

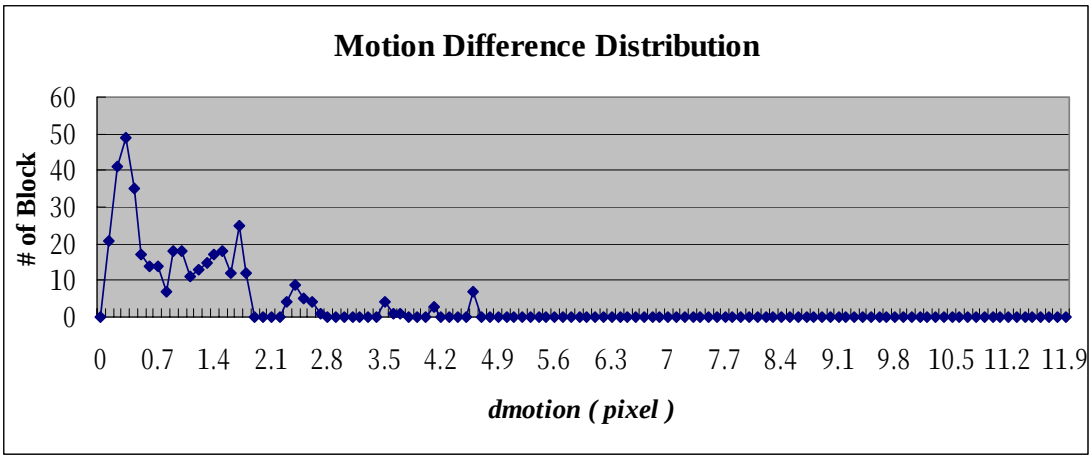


(k) 參考 Frame 8 所求得的區域運動向量

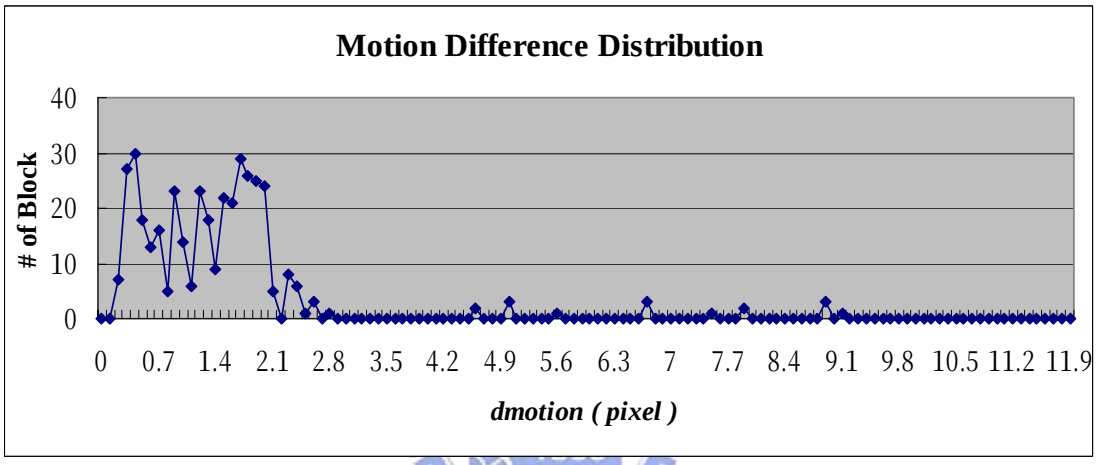


(l) 參考 Frame 8 所求得的全域運動向量

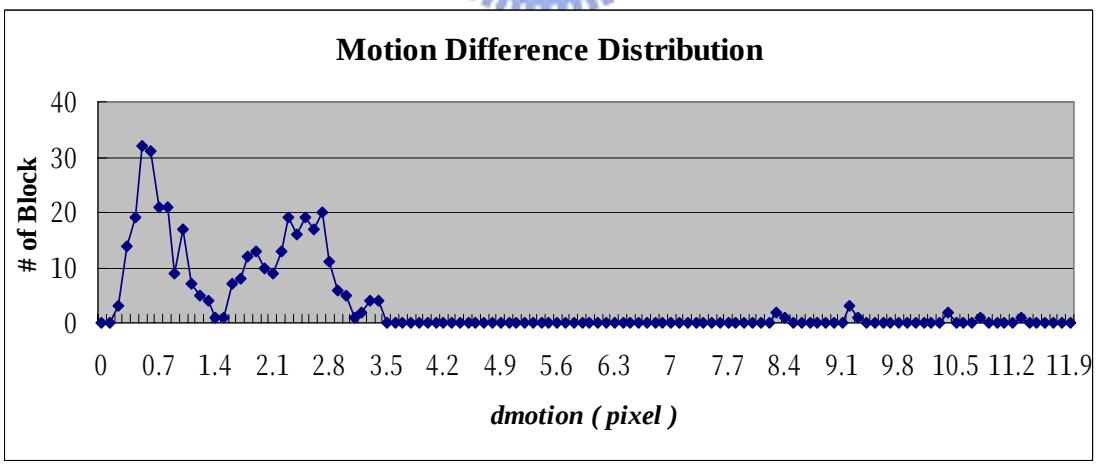
圖 49：使用不同參考畫面所求得的區域運動向量與全域運動向量



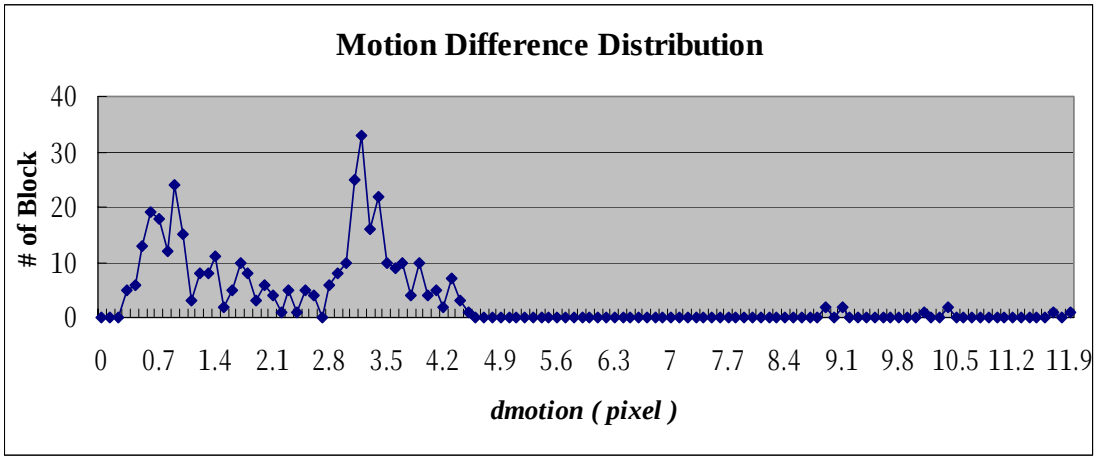
(a) 參考 Frame 1 所求得的 d_{motion}



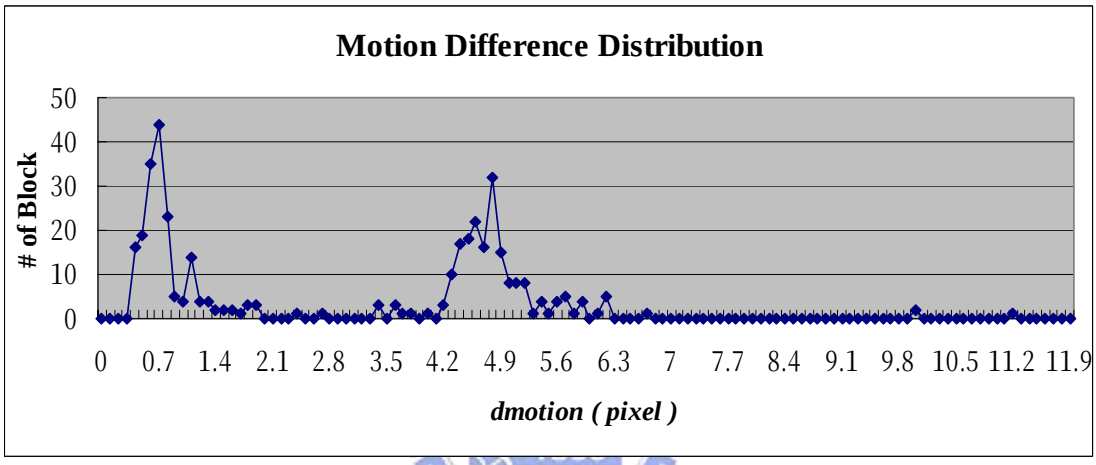
(b) 參考 Frame 2 所求得的 d_{motion}



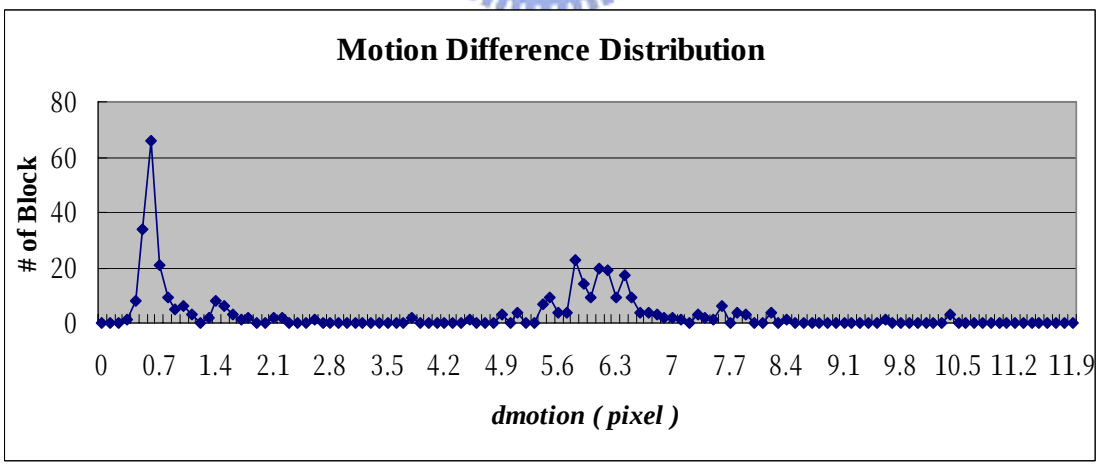
(c) 參考 Frame 3 所求得的 d_{motion}



(d) 參考 Frame 4 所求得的 d_{motion}



(e) 參考 Frame 4 所求得的 d_{motion}



(f) 參考 Frame 8 所求得的 d_{motion}

圖 50：使用不同參考畫面所求得的 d_{motion} 分佈

(a) 參考 Frame 1 所得到的切割結果

(b) 參考 Frame 2 所得到的切割結果

(c) 參考 Frame 3 所得到的切割結果

(d) 參考 Frame 4 所得到的切割結果

(e) 參考 Frame 6 所得到的切割結果

(f) 參考 Frame 8 所得到的切割結果

圖 51：使用不同參考畫面之自動切割結果

表 16：使用不同參考畫面時的切割正確率（共切割 50 張畫面）

參考畫面	Frame 1	Frame 2	Frame 3	Frame 4	Frame 6	Frame 8
切割正確率	67.01%	69.9%	74.71%	85.22%	90.82%	93.06%
錯誤的背景	32.23%	29.48%	23.97%	13.05%	7.23%	5.26%
錯誤的前景	0.76%	0.62%	1.32%	1.73%	1.95%	1.69%
區域運動向量在 1 以下佔全體比例	98.374%	78.965%	16.26%	8.67%	7.31%	5.14%

表 16 列統計了 50 張 Mobile 畫面使用本論文提出的自動切割方法之結果，切割流程中的全域運動向量均是使用 loop10 的攝影機運動參數求出，而 d_{motion} 之臨界值設定為 2。表 16 中切割正確率的判斷是使用人眼觀察，由於切割的時候是以區塊為單位，因此當某一個區塊包含前景區域，此區塊即被判斷為前景，反之則為判斷為背景。在進行切割的時後，當參考畫面足夠遠時，畫面中的背景部分與前景部分之運動趨勢差異夠大，則切割的正確率會比較好，由於 PTZ 攝影機運動參數模型有兩個像素的誤差，因此當前景部分與背景部分的運動趨勢差異太小時，會有切割錯誤的情況。表 16 中，“錯誤的背景 ” 欄位是指原本是前景，但是被切割成背景，被切割成背景的原因是因為前景部分的運動與攝影機運動差異太小，小於兩個像素內才會造成錯誤的切割。“錯誤的前景 ” 欄位是指原本是背景，但是被切割成前景；不管參考畫面的遠近，d_{motion} 之臨界值設定為 2 能夠很穩定的把背景區域切割出來。

進行區塊比對求得區域運動向量時，所得到的運動向量均為整數像素值，當畫面中的運動越小，或前景與背景的運動越相似時，運動向量的精準度越為重要，另外當運動越小時，所求得的攝影機參數亦會不夠精確。表 16 中最後一個欄位列出了不同參考畫面之區域運動向量小於 1 佔全體的比例，當小於 1 的比例太

多，精確度的影響就更為重要，因此，如果運動向量估計的不精確，就會造成較差的切割結果。從表中可以看到當參考畫面由 2 改為 3 時，運動向量小於 1 的比例大幅減少，而從參考畫面 3 改為參考畫面 4 時，運動向量小於 1 的比例減少了約兩倍，而參考畫面為 4、6、8 時，隔了兩張畫面，減少的比例不到 2%，已經有不錯的切割結果，因此當運動向量小於 1 的比例非常少，並且只有小幅度的減少時，受到運動向量精確度不足的影響就會變小，也會有較好的切割結果，則此時選擇的參考畫面是合適的。

表 17 中列出切割一張畫面所佔的時間為 3452 ms，其中區塊比對的時間佔了大部分，共花了 2876 ms，而修正偏差的攝影機參數共花了 556 ms。由於視訊壓縮時與切割時的區塊比對是相同的，因此若能整合在一起，也可以節省重複花費的時間。

表 17：切割時間

切割時間	3452ms	區塊比對的時間	2876ms	更正攝影機運動參數的時間（loop10）	556ms

第六章 結論與未來展望

本篇論文使用的編碼方式是把影像分成前景和背景來討論，希望依照壓縮的需求以不同的編碼方式來達到編碼效率的改善。在此編碼效能之改善為 1.前景的部分 PSNR 品質的提升、2.背景的部分減少資料壓縮量與 3.整體壓縮時間減少，針對這些部分，利用到區塊大小和 Rate-Distortion 的參數值設定，的確可以有效改善前景的畫面品質和背景的資料量。

由於整個 H.264 標準的架構非常龐大，可以再分析的部分也很多，如量化參數、參考畫面數、Entropy 編碼等，都可能繼續應用在前景和背景的壓縮。在本篇論文，我們都以符合標準的解碼程序來做研究，當然，如果能在標準規定的語法和語意外來進行編碼，前景和背景分析的結果，應該會有更大的空間發展。

在本論文中亦提出了一個自動切割的方法，從實驗顯示，本方法可以切割出與攝影機運動不同的物體，比較可能切割錯誤的地方在於畫面中一大片的平滑區域，造成的原因如 5.3 小節所敘述，乃是因為區塊比對誤差所造成，而使用光流估測的方法亦會有同樣的缺點，因此區域運動估測與對應點的改善是一個重要的課題。另外，本論文所提出的方法可以切割出前景與背景兩個區域，考慮其他的應用下，可以再詳細對不同運動的物體做切割，例如將 Mobile 影片中的月曆、球與火車各自切割，此為後續可研究的方向之一。由於大多數的壓縮標準均以區塊為壓縮單位，所以以區塊為基礎的切割方法中，可以適用於多數的壓縮標準，若考慮以 MPEG4 之 VOP [8]為壓縮單位，可以考慮使用本論文的方法在切割好的物體之中，再針對每個物體的輪廓做細部切割。本論文亦可當成其他視訊切割方法的基礎，例如，只切割一張影像並找出獨立運動的物體當成初始切割，之後可能使用物體追蹤之方式，繼續對不同的畫面做切割。

參考文獻

[1] ITU-T Rec. H.264 / ISO/IEC 11496-10, “Advanced Video Coding,” Final Committee Draft, Document JVT- E022, September 2002.

[2] ITU-T Rec. H.264 / ISO/IEC 11496-10, “Advanced Video Coding,” Final Committee Draft, Document JVT-F100, December 2002.

[3] Iain E G Richardson, “H.264 and MPEG-4 Video Compression,” *John Wiley & Sons, 2003.*

[4] ITU-T Rec. H.264 / ISO/IEC 11496-10, “Advanced Video Coding,” Final Committee Draft, Document JVT-G050, March 2003

[5] G. J. Sullivan and T. Wiegand, “Rate-Distortion Optimization for Video Compression,” *IEEE Signal Processing Magazine*, vol. 15, no. 6, pp. 74–90, Nov. 1998.

[6] T. Wiegand, H. Schwarz, A. Joch, F. Kossentini and G. J. Sullivan, “Rate-Constrained Coder Control and Comparison of Video Coding Standards,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no.7, pp. 688 – 703, 2003.

[7] M. Flierl and B. Girod, “Generalized B Pictures and The Draft H.264/AVC video-compression standard,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13 , no. 7 , pp.587 – 597, 2003.

[8] ISO/IEC IS 11172-2, MPEG-1 Video.

[9] ISO/IEC IS 13818-2, MPEG-2 Video.

[10] ISO/IEC IS 14996-2, MPEG-4 Video.

[11] JVT Reference Software, ftp://ftp.imtc-files.org/jvt-experts/reference_software/ .

[12] R. C. Gonzalez and R. E. Woods, “Digital Image Processing,” *Prentice Hall*, 2 edition, 2002

[13] M. Kass, A. Witkin, and D. Terzopoulos, "Snakes: Active Contour Models," *Int. Journal of Computer Vision*, vol. 1, pp. 321~331, 1988.

[14] L. Vincent and P. Soille, "Watersheds in digital spaces: an efficient algorithm base on immersion simulations," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, pp. 583~598, 1991.

[15] J. Shi and J. Malik, "Normalized Cuts and Image Segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 8, Aug. 2000.

[16] R. Castango, T. Ebrahimi and M. Kunt, "Video Segmentation Based on Multiple Features for Interactive Multimedia Application," *IEEE Transactions on Circuits and Systems for Video Technology,* vol. 8, no. 5, Sept. 1998.

[17] S. Y. Chien, S. Y. Ma, and L. G. Chen, "Efficient Moving Object Segmentation Algorithm Using Background Registration Technique," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 12, no. 7, July 2002

[x] R. Mech and M. Wollborn, "A noise robust method for segmentation of moving objects in video sequences," *Proc IEEE Int. Conf. Acoustics, Speech, Signal Processing, ICASSP'97*, Munich, Germany, Apr. 1997, vol. 4, pp. 2657–2660.

[18] G. Adiv, "Determining three-dimensional motion and structure from optical flow generated by several moving objects," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. PAMI-7, pp. 384–401, July 1985.

[19] B. J. Frey, N. Jojic, and A. Kannan, "Learning Appearance and Transparency Manifolds of Occluded Objects in Layers," *Proc. Computer Vision and Patten Recognition*, vol. 1, pp. 45–52, Madison, WI, 2003.

[20] J. Shi and J. Malik, "Motion Segmentation and Tracking Using Normalized Cuts," *IEEE Int. Conf. Computer Vision*, pp. 1154-1160, 1998.

[21] T. Meier and K. N. Ngan, "Automatic Segmentation of Moving Objects for Video Object Plane Generation," *IEEE Transactions on Circuits and Systems for*

Video Technology, vol. 8, no. 5, Sept. 1998.

[22] J. G. Choi, S. W. Lee, and S. D. Kim, "Spatio-Temporal Video Segmentation Using a Joint Similarity Measure," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 7, no. 2, April 1997.

[23] R. V. Babu, K. R. Ramakrishnan, and S. H. Srinivasan "Video Object Segmentation: A Compressed Domain Approach" *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 14, no 4, pp. 462-474, 2004.

[24] Z. G. Li, F. Pan, K. P. Lim, G. N. Feng, X. Lin, S. Rahardia, D.J. Wu, "Adaptive frame layer rate control for H.264,"*IEEE Int. Conf. Multimedia and Expo*, vol. 1, pp.581–584, 6–9 July 2003.

[25] Y. Lu, W. Gao, and F. Wu, "Efficient Background Video Coding With Static Sprite Generation and Arbitrary-Shape Spatial Prediction Techniques," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 13, no 5, pp. 394-405, 2003.

[26] F. Pan, X. Lin, S. Rahardja, K. P. Lim, Z. G. Li, D. Wu, and S. Wu "Fast Mode Decision Algorithm for Intraprediction in H.264/AVC Video Coding" *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 15, no 7, pp. 813-822, 2005.

[27] Z. K. Lin, H. H. Lin, Y. H. Chen, and J.H. Chuang, "On The Performance Improvement Of H.264 Through Foreground and Background Analysis" *IEEE Int. Conf. Multimedia and Expo*, 6–8 July 2005.

[28] B. K. P. Horn and B. G. Schunck, "Determining Optical Flow," *Artif. Int.*, vol. 17, pp. 185–203, 1981.

[29] Y. P. Tan, S. R. Kulkarni, and P. J. Ramadge, "A New Method for Camera Motion Parameter Estimation," *Proc. IEEE Int. Conf. Image Processing*, vol. 1, pp. 406–409, Oct. 1995.

[30] Y. P. Tan, D. D. Saur, S. R. Kulkarni, and P. J. Ramadge, “Rapid Estimation of Camera Motion from Compressed Video with Application to Video Annotation,” *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 10, no. 1, Feb. 2000.