國立臺灣大學電機資訊學院資訊網路與多媒體研究所

碩士論文

Graduate Institute of Networking and Multimedia

College of Electrical Engineering and Computer Science

National Taiwan University

Master Thesis

# 基於系統狀態下之可調適影音播放器

# Linux based manageable MPEG-4 applications with resource monitor

鄭時旭

Shih-Hsu Cheng

指導教授：陳文進 博士

Advisor: Wen-Chin Chen, Ph.D.

中華民國 97 年 6 月

June, 2008

# 口試委員審定書

# 誌謝

這篇論文之所以能夠完成，首先要先感謝陳文進教授在這兩年中的諸多教誨與協助，感謝劉登榮學長在研究方向上以及完成論文的過程中所給予的指導與建議，感謝各位學長們，周賢德、邱盈創、林豐毅、樂臺裔，在我剛進研究所時的照顧以及對於課業及學習上的分享，感謝我的同學們，陳冠中、陳俊綱、莊秉原，在課業、生活以及研究上的幫助與陪伴，感謝學弟，吳瑞欣，提供我在寫論文時所需要的幫忙，還要感謝實驗室所提供的良好研究環境與設備，讓我能順利完成研究，最後，要感謝我的父母與家人，在我求學的過程中所給予的支持與鼓勵，尤其是這一兩個月早出晚歸，總是關心我的身體及進度，讓我能很順利的完成學業，謝謝。

鄭時旭 謹誌

民國九十七年六月

# 摘要

由於嵌入式系統與手持裝置的日漸普及，系統資源不足的問題也越來越被重視。在這些架構下所能夠提供的資源以及運算量較一般系統低，由於資源不足或是運算量的缺乏而產生的問題將導致系統的不穩定，在無法有效的提供更大量的資源及運算量情況下，將必須以結束或暫時停止程序來維持系統穩定。

放棄或暫時停止程序對於所排定的工作進度會有一定的延遲，同時也影響系統提供的功能，如果能以調整某部分工作效能的方式取代放棄或暫時停止程序的方式來讓系統回歸到穩定的狀態運作，提供使用者在系統資源分配不均或運算量過高的情形下也能夠使用到相同的功能。

在本篇論文中提出一個在 Linux 環境下的多媒體播放架構，目的是在系統資源不足而呈現不穩定的狀態下，播放器能依照資源監視器所提供的資訊調整播放的模式以及以減少解碼工作的次數來降低所需要的運算量，同時持續提供使用者播放影片的功能。當系統資源使用量降低後，播放器會將模式轉換回一般的播放方式讓影片有較好的呈現。藉由調整播放器模式的方法，使系統遇到資源不足所導致的不穩定狀況時，系統依然能夠提供使用者完整的系統功能。

關鍵詞：多媒體播放、資源監視、資源分配、系統穩定

# Abstract

The problems of system resources are more important due to the growing popularity of embedded systems and handheld devices. These systems can’t provide resources and computing power as much as those that normal systems can. Lack of resources or computing power will make these systems instable. We should suspend or abandon some process in order to make the system back to a stable state.

Abandon or suspend functions will let scheduled progress delay and make users feel uncomfortable. We hope that we can find a way to make system stable with adjusting computing or resource usage of player in stead of abandonment or halt of process.

In this thesis, I have designed and implemented a framework of multimedia player based on Linux. Multimedia player will switch mode to low computing mode when there are not enough system resources. By reducing the number of decoding frames in the video, the player decreases the CPU usage in this mode. Player will automatically adjust the mode.

Key words: Multimedia player, System resources, System stable

# 內容目錄

# 圖目錄

# Chapter1.緒論

## 1.1 背景

### 1.1.1 監視系統

一個簡單的監視系統裡具有幾個重要的功能，其中包括影像擷取、影像儲存、以及影像播放三個部分。影像擷取由攝影機負責接收監視區域的影像，收集到的資訊代表監視區域目前所顯示的情形。當攝影機接收到影像資料後，監視系統會將資料經過編碼後存進硬碟中，透過這個儲存的動作，系統可以提供使用者重新播放、倒轉、快轉、暫停等功能以方便對於監視區域資訊的掌握。即時的影像播放則是提供使用者可以透過螢幕看到目前攝影機所收集到的影像，以便隨時對發生的變化做出反應。

在這裡提出一個監視系統的範例來說明對於一個監視系統在系統資源的使用以及分配上會發生的問題。

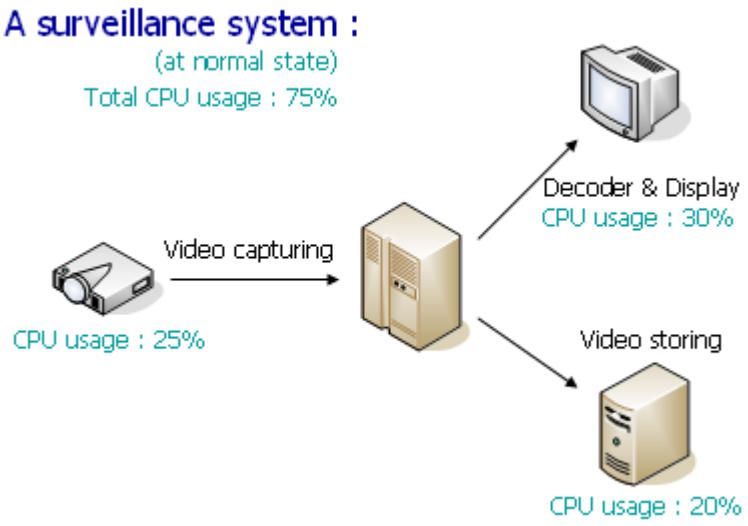



圖 1-1　監視系統一般運作情形

假設在穩定運作的狀態下，系統能夠提供的最大的 CPU 使用量爲 80%，目前系統中主要有三個工作需要使用到 CPU，其中影像擷取的部分需要 25%，影像儲存的部分則是需要 20%，最後即時解碼及播放影片有 30%的需求，所以要完成這三個工作，系統總共需要提供 75%的 CPU 資源，而系統在穩定狀態下所能夠提供的最大 CPU 使用量爲 80%，表示在一般進行這三個工作時，整個系統是處於一個穩定的狀態。

### 1.1.2 非預期資源使用

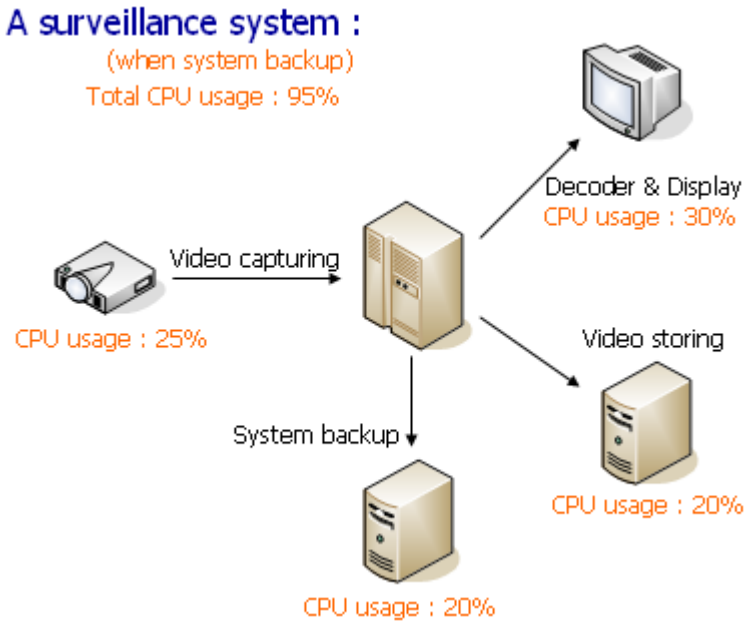



圖 1-2　監視系統於系統備份下之運作情形

當系統出現一些非預期的資源使用，例如對舊影片進行備份的動作，CPU 的使用率以及系統的穩定度就與進行一般工作時不一樣。影像接收、影像儲存、以及影像播放三個動作都是持續在進行的，如果說這些工作都維持一樣的效能時，影像擷取使用 25%，影像儲存則使用 20%，影像播放也還維持 30%的使用量，但是新增了一個備份影片的動作，假設需要 20%的 CPU 運算。如此一來整個系統的 CPU 使用量達到 95%，如果讓系統一直維持在這麼高的 CPU 使用率會造成整

個系統呈現一個不穩定的狀態。

## 1.2 動機

系統穩定性一直是在系統以及應用程式在設計和運作上很重要的課題，系統的不穩定可能造成系統效能降低、應用程式錯誤、或者導致系統停止等情形。導致系統不穩定有很多原因，像是運算量過大系統無法負荷、資源分配不均或是資源不足都會導致系統不穩定。

嵌入式系統以及手持裝置日漸普及，提供服務的系統供應者提出在嵌入式系統及手持裝置上提供與一般電腦上相同的服務以及工作。嵌入式系統與手持裝置上所能提供資源以及運算能力和一般電腦或系統比較起來是少了許多。所以更容易在系統運作時遇到資源分配不均或者是運算能力不足而導致整個系統在一個不穩定的狀態下運作。

如前小節所提到監視系統的例子，若出現像系統備份這種非常態性的工作或者是資源使用，就必須選擇其中一個工作進行效能及系統資源使用上的調整，並且持續提供能被使用者所接受的系統功能，非預期的資源使用持續時間越短，就表示系統能夠更快的回復到一個穩定的狀態，所以不對於系統備份的工作進行效能調整。

觀察監視系統本身的功能，影像擷取負責處理接收攝影機資訊，是整個系統的資訊的來源，如果降低這個部分的效能有可能會造成整個系統資訊的不完整，所以不對這個部分進行調整。再來看看影像存取的功能，由於所接收到的影像都必須經過編碼存進硬碟中，提供使用者可播放的影像資料。影像擷取與影像儲存

是兩個必須共存的功能，先由影像擷取獲得資訊再提供給影像儲存的部分進行編碼及儲存，影像儲存提供使用者察看不同時間點上面影片出現的資訊，調整影像儲存的效能會造成無法提供使用者正常察看影片的情況，所以我們決定調整影像播放的部分。在一個監視系統中，所收到及錄製的影像與日常生活中的所看到的影片需要良好的畫質以及很高的 frame rate。因此我們就決定改良這個部分來進行整個架構的設計。

## 1.3 目標與貢獻

在本篇論文中提出一個在 Linux 作業系統上 MPEG-4 應用程式的架構來解決非預期資源使用所產生系統不穩定的問題，其中這個架構由以下的一些功能所組成：

- 用來收集系統資源使用情形的資源監視器
- 可依資源監視器所提供的資源使用情形來調整播放模式的播放器
- 播放器在不同的模式下調整資源的使用量

在此以資源使用的示意圖(圖 1-3)來表示在本篇論文所提出的架構下，希望達成的目標。曲線圖中包括整個系統遇到非預期的資源使用、以及調整播放器模式後，系統CPU使用量的情形。首先我們設定 80%爲系統穩定的臨界點，當系統處於一般狀態運作下的CPU使用率在 70~80%跳動。當碰到非預期的CPU使用(圖 1-3 的A)，如之前提到的監視系統備份工作，CPU的使用量提高至 95%，已經超過我們設定系統可接受的上限值，如果這個狀態持續下去會造成系統的不穩定，播放器從資源監視器獲得目前系統CPU使用的情形，而自動進行模式的調整來減低其運算量進而達到整個系統的CPU使用量下降(圖 1-3 的B)，讓系統回到一個穩定的狀態，所有工作都能持續進行。當系統備份結束後會釋放原來所使用的資源，讓整個系統的CPU使用量降低，此時播放器會再將模式調回一般播放的狀態，提供

較好的播放效率(圖 1-3 的C)。

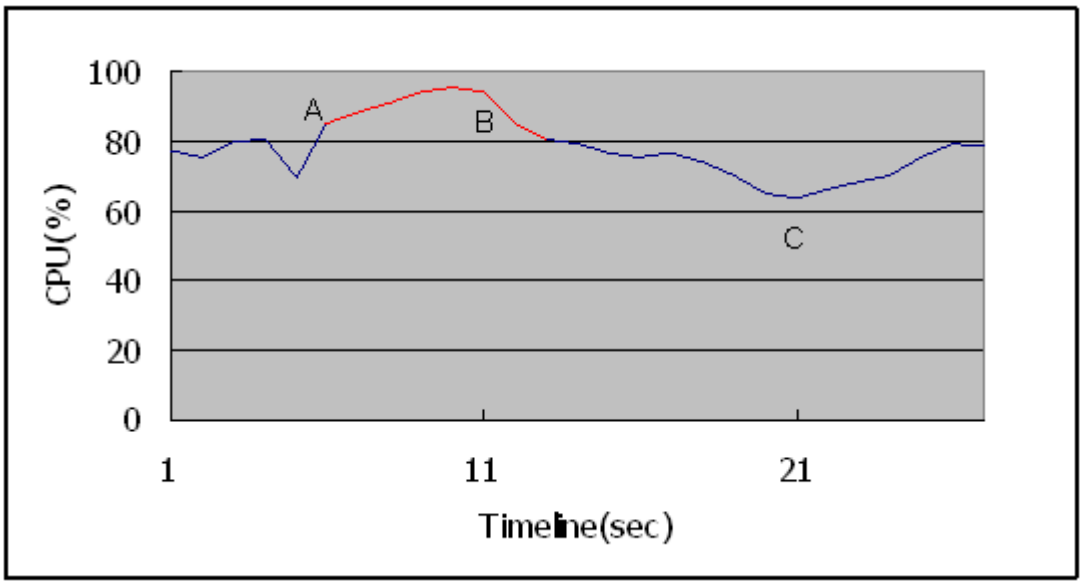



圖 1-3　　系統資源使用情形示意圖

# 1.4 章節介紹

接下來的部分，第二章介紹已存在的工具及提供資源監控及資源控制的架構，於第三章說明整個系統架構的想法和實際作法，其中包括系統資源監控及運算量的控制方法，在第四章的提出本篇論文的總結以及未來可以進行改善的工作。

# Chapter2.相關技術

## 2.1 相關架構

劉又誠學長在 94 年所發表的論文：Resource Monitor and Decoder Control API for MPEG-4 Application 中，提出了一個架構來解決在系統資源不足的狀態下，透過資源監視以及資源調整的方式來完成多個影片的播放。其中這個架構中較爲重要有三個部分：MPEG-PY 系統、資源監視器、及資源管理器。

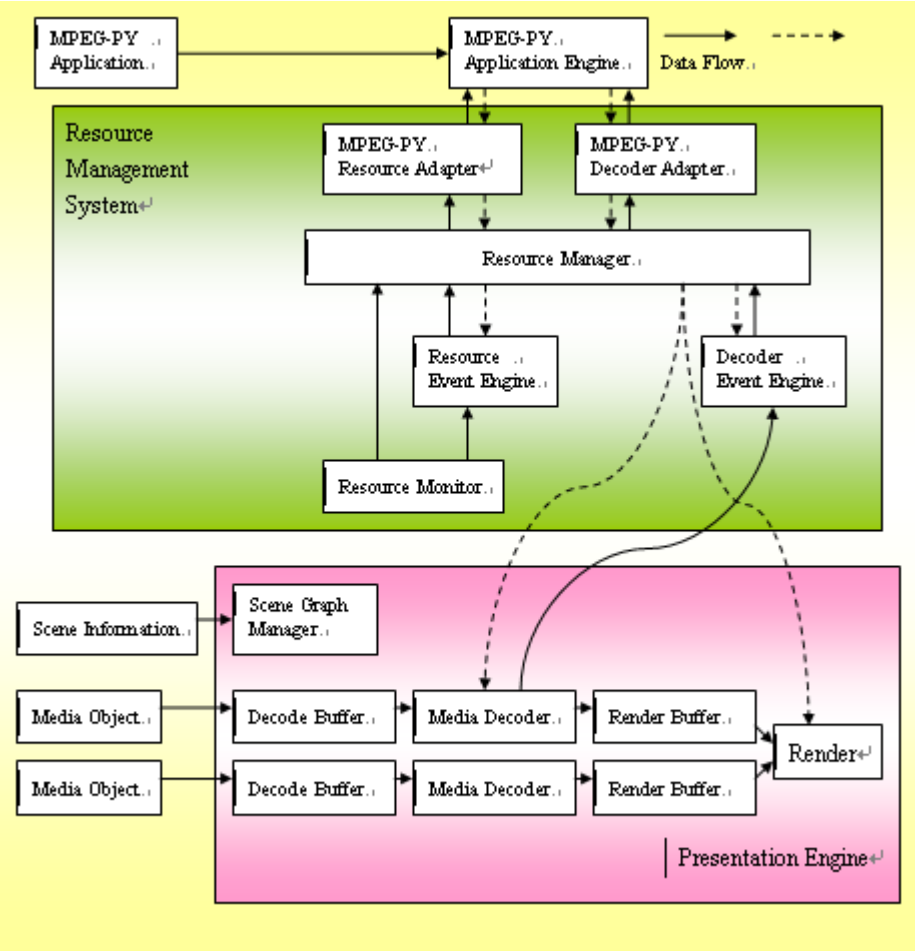



圖 2-1　　資源監視與控制器之整體架構圖

### 2.1.1 MPEG-PY 系統

MPEG-PY 系統提供一套由 Python 語言所撰寫的工具提供開發者去建立出各種 MPEG-4 的應用程式，播放器以 MPEG-PY 作爲主要架構進行實作，下圖爲 MPEG-PY 的系統架構圖。

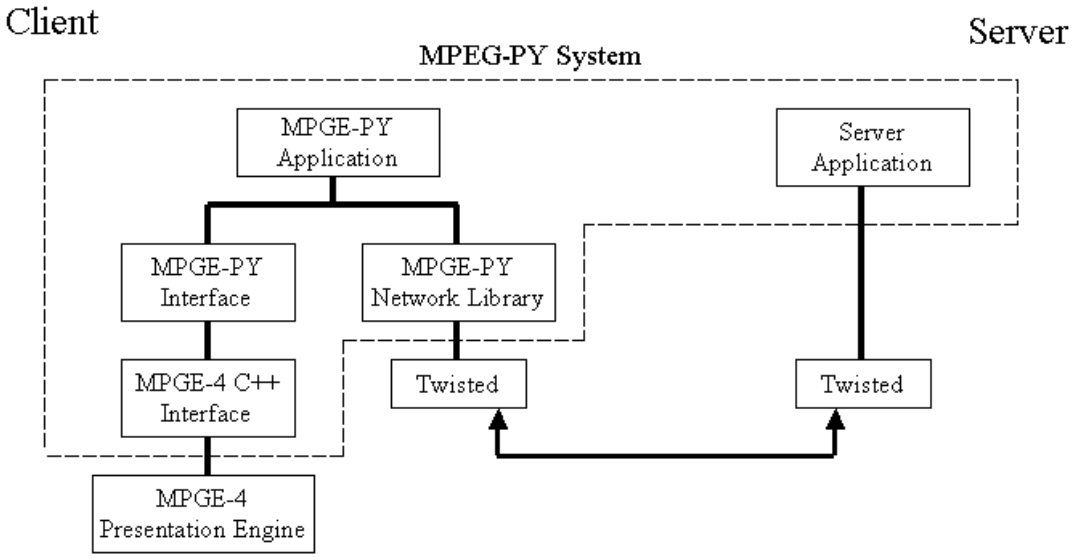



圖 2-2 MPEG-PY 架構圖

.

### 2.1.2 資源監視器

資源監視器是以微軟平台上的 Performance Data Helper（PDH）作爲基礎。 PDH 是一套用來建立效能監視器的一套應用程式。使用這套介面在建立系統時需要與 PDH.dll 作連結。

PDH 包括三個部分：效能物件與計數器、PDH 查詢、以及效能資訊來源。

- 效能物件與計數器：

系統效能的資訊由效能計數器進行收集，而這個計數器名稱會存在系統的登錄檔中。每一個效能計數器都會對應到系統中的一個工作，且有各自的名稱及路

徑。絕大多數的效能計數器數值都是遞增的而且永遠不會歸零。

- PDH 查詢：

PDH 函式透過查詢與計數器來運作。查詢是將效能計數器統整，這樣就可以同時收集群組內計數器的資料。一個查詢可以包含一個或多個效能計數器，而一個計數器也能同時存在於多個查詢群組中。

- 效能資訊來源：

許多 PDH 函式收集系統效能資訊的來源包括兩個：即時數據來源和效能紀錄檔。即時數據來源提供關於目前系統中效能活動的資訊，而效能記錄檔的部分是儲存成二元或是文字檔的格式，其中包含了效能資訊的數據.以及提供這些數值的計數器名稱。

系統效能資訊隨著時間不斷的變化，如果爲了取得某個時間點上的系統效能資訊，而讓系統等待直到完成收集的動作，會讓目標的應用程式有所延遲，所以在設計上資源監視器由一個執行緒來負責，資訊來源的選擇上，即時數據來源是比較好的選擇。

### 2.1.3 資源管理器

資源管理器連接了資源監視器與 MPEG-PY 兩個部分。一方面從資源監視器接收系統資源使用狀態的資訊，另一方面依照資源監視器傳回來的資訊將調整的參數傳給 MPEG-PY，以調整播放器所使用的系統資源。

資源管理器在判斷調整方式主要分爲三個階段進行調整：

- 第一階段：

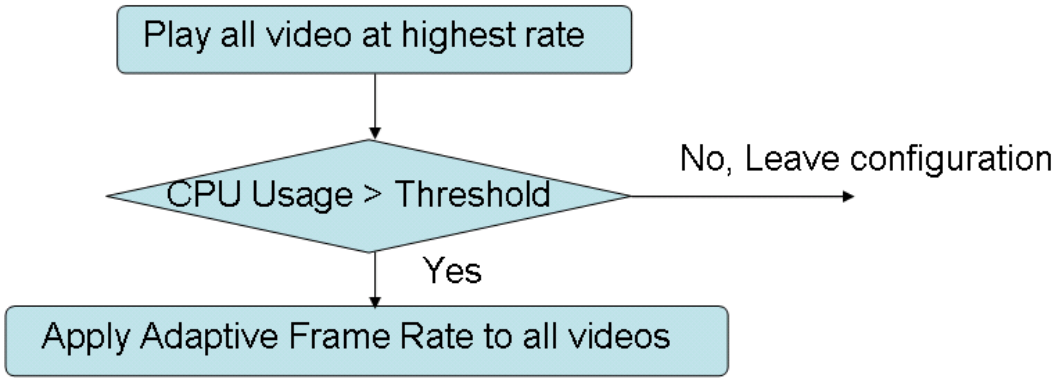



圖 2-3　第一階段判斷 CPU 與上限值關係

初始狀態下，所有的影片都用最高的效率進行播放，如果目前的播放方式所需要的 CPU 使用量沒有超過我們所設定的資源使用的門檻，就繼續依照這個模式進行播放。如果超過我們所設定的上限值，則進行下一個階段的調整。

- 第二階段：

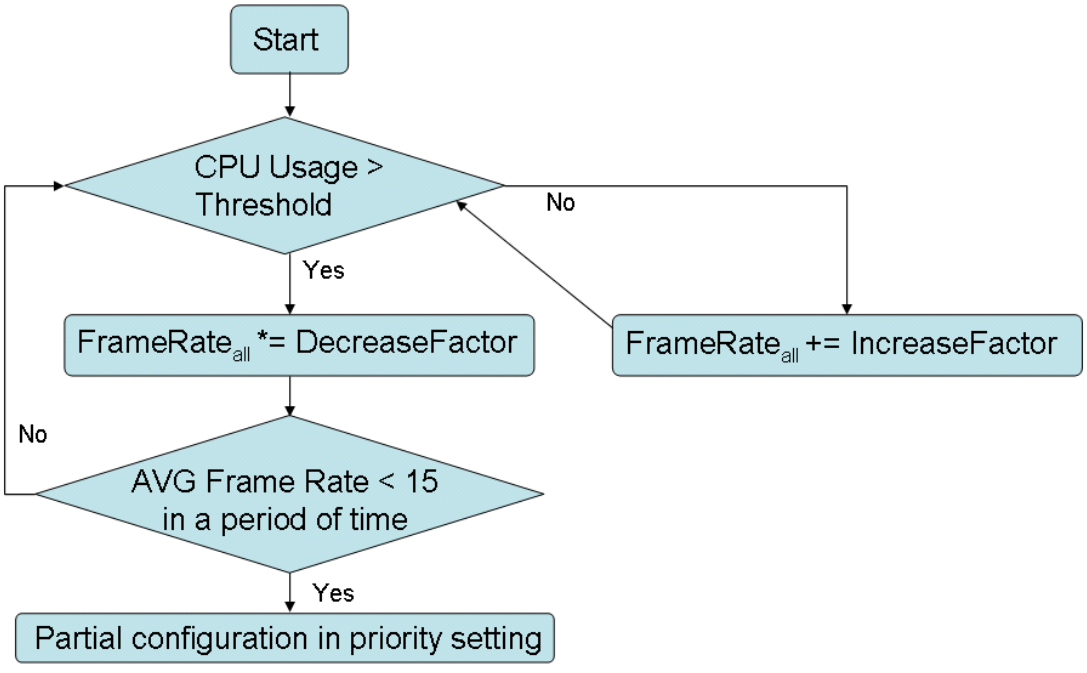



圖 2-4　第二階段依照參數調整各影片 frame rate

在這個階段同一個場景中所有的影片都套用同一個調整 frame rate 的演算

法，初始階段進行影片播放時，每一個影片在各自的 frame rate 以最好的畫質進行播出。當 CPU 使用量超過設定的上限時，每個影片依照所設定的下降參數來進行 frame rate 的調整以達到降低 CPU 使用量的效果。每個影片的 frame rate 下降幅度都很快，如此一來可以讓 CPU 的使用量快速的降到所設定的門檻以下。在 CPU 的使用量比資源使用設定的上限要低時，影片的 frame 會依照不同的上升參數慢慢的提升回來。

在這個階段中，降低 frame rate 的幅度會有一個最低限度。一些研究指出，如果每秒播放的 frame 數比 15 小會讓影片播放延遲的情形很嚴重。而不幸的是人的眼睛對於這種的播放延遲會有很明顯的反應，所以當 CPU 使用率一直處於很高的狀態超過一段所設定的時間後，我們就進行下一個階段的調整。

- 第三階段：

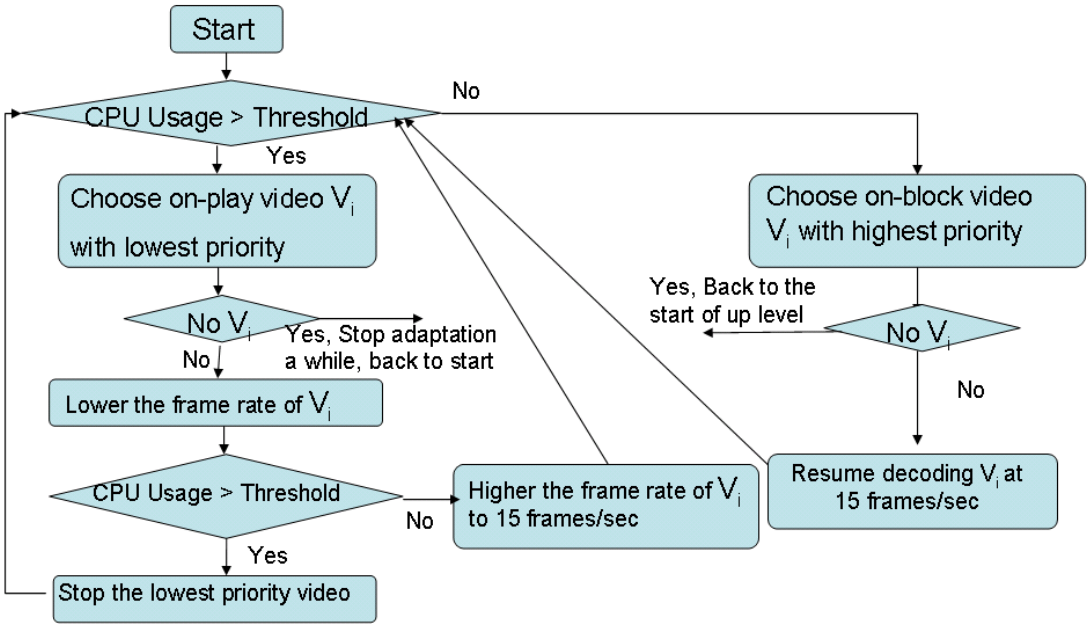



圖 2-5　第三階段依照優先權逐步調整 frame rate 及停止影片播放

在第三階段所進行的是對於部分的影片來進行效能的調整，在整個場景中挑選出優先權最低的影片，將這個影片的 frame rate 往下調整到 15fps 以下，如果 CPU 的使用量還是沒辦法比設定的上限值小，資源管理器就會停止解碼這個影片。然後一直重複這個動作直到 CPU 使用量是符合我們所設定的上限值。

如果 CPU 使用量小於上限值時，對於播放 frame rate 比 15fps 小的影片會有 frame rate 還原的一個動作，資源管理器會將這些本來因爲 CPU 使用量過大而被迫調整 frame rate 的影片作一個調整，將這些影片的 frame rate 往 15fps 進行移動。

對於被暫時停止播放的影片而言，若 CPU 的使用量小於設定的上限值，而所有已經在播放的影片 frame rate 也都超過 15 時，資源管理器會在這些暫時停止播放的影片中找一個優先權最高的影片來進行回復，這個回復的動作在 CPU 使用量比設定的上限值低的時候會持續的進行。

若在整個場景中已經沒有暫停播放的影片，且每個影片的 frame rate 都超過 15 時，資源管理器會回到第一階段進行監控。

# 2.2 MPEG-4

## 2.2.1 MPEG-4 介紹

MPEG-4 是 MPEG(Moving Picture Expert Group)這個組織所定出來的一個標準。主要是應用在數位電視、互動式的圖片的應用程式以及網路多媒體上，不同於 MPEG-1 及 MPEG-2，MPEG-4 爲了使聲音及影像達到高品質、體積小、應用層面更廣的要求，於是導入了物件的概念，將影像及聲音物件化。

MPEG-4 主要涵蓋的範疇分爲三個部分：聲音及影像的壓縮、傳遞、及互動。

在多媒體壓縮的部分，由於行動裝置的普及，MPEG-4 的目標希望能夠提供一個 bit rate 較低且具有良好的品質。編碼過後的多媒體可以以較低頻寬的方式儲存在較小的儲存裝置中。

MPEG-4 在傳遞上提出一個虛擬的多媒體傳遞介面 DMIF (Delivery Multimedia Integration Framework)，無論是對於檔案播放或是在網路上傳遞，都包含在這個介面內。

有別於傳統的影音缺乏與觀賞者之間的互動，在數位影音加上互動的內容成爲一種潮流，MPEG-4 提供許多工具來建立一個動態且可以互動的場景：包括對於利用虛擬聲音的物件來完成場景的描述、與動態的場景進行一些互動(如對場景中的物件作點選後將會出現動畫)、以及可程式化的機制讓整個互動性更豐富。

### 2.2.2 FFMPEG

FFMPEG 是一套發展完全，具有錄音、轉換及播放多媒體的工具。其中包括對負責提供聲音及影像解碼的函式庫：”libavcodec”。FFMPEG 是建立在 Linux 系統下，不過現在可以在大部分的作業系統下進行編譯，其中也包括 Windows。

FFMPEG 這套工具中包含了以下幾個部分：

- Ffmpeg 是一個命令列上的工具，提供將影片轉換檔案格式的功能。這個工具也支援電視卡上訊號的即時擷取及編碼。
- Ffserver 是一個 HTTP(RTSP 也在建構中)上即時廣播的多媒體串流伺服器。
- Ffplay 是一個以 SDL 及 FFmpeg 函式庫爲基礎的多媒體播放器。

- Libavcodec 是一個包含所有 FFMPEG 中所使用之解碼器及編碼器的函式庫，絕大部分的 codec 都能有好的效率以及較高的重複使用性。
- Libaformay 則是負責解析處理及建立各式各樣常見的聲音 / 影像檔案標頭檔的函式庫。

# Chapter3.系統設計與實作

## 3.1 系統架構簡介

在 FFMPEG 播放的架構裡，影像及聲音的每個 packet 會分別排進 audio queue 以及 video queue 兩個序列中，依照進入序列的先後順序以及 frame 中 decode timestamp 的數值來進行解碼。

由於處理聲音的部分相較於處理影像需要較少的資源及運算量，所以在本篇論文中，只對影像的部分進行處理。從 video frame queue 中取出 frame 的資訊後，對 frame 進行解碼的工作是整個播放器中最需要系統資源的部分，在我們所設計的系統架構中，當系統處於高負載狀態時，播放器透過對 video frame queue 中的 frame 進行篩選後降低所要解碼的 frame 數量，進而降低播放器所需要的系統資源。

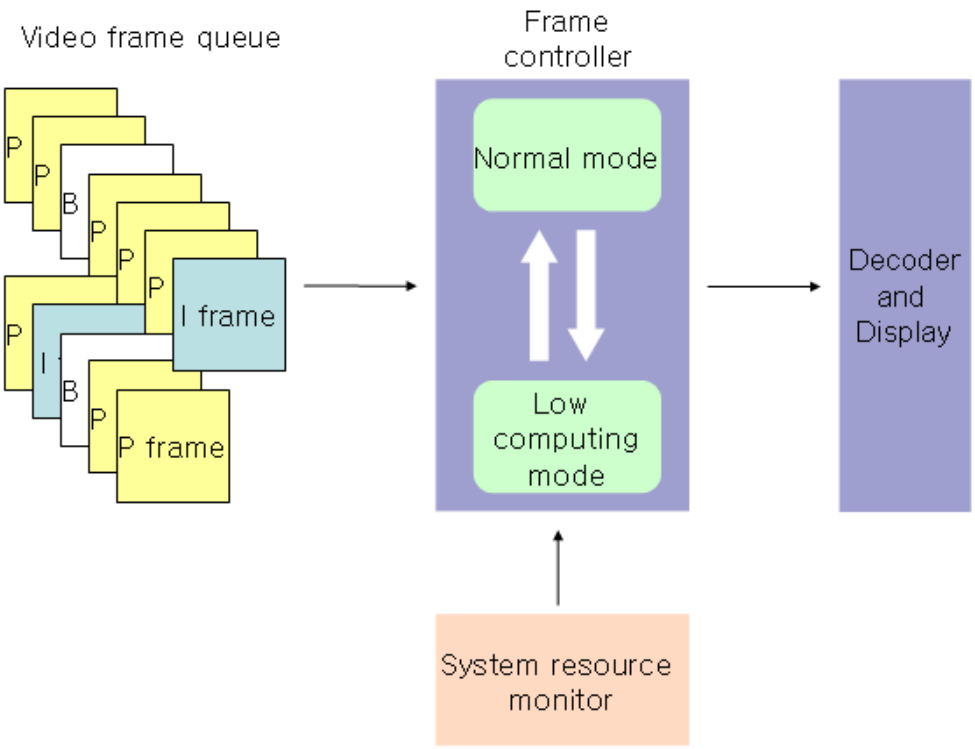



圖 3-1　　系統架構圖

因此我們從 video frame queue 中取出 frame 傳給解碼器解碼前，加入了「resource monitor」以及「frame controller」兩個函式，一方面取得系統資源的使用資訊、一方面透過這兩個函式來調整播放器的模式及決定 video frame queue 中哪些 frame 是要被留下來的，哪些則是需要捨棄。

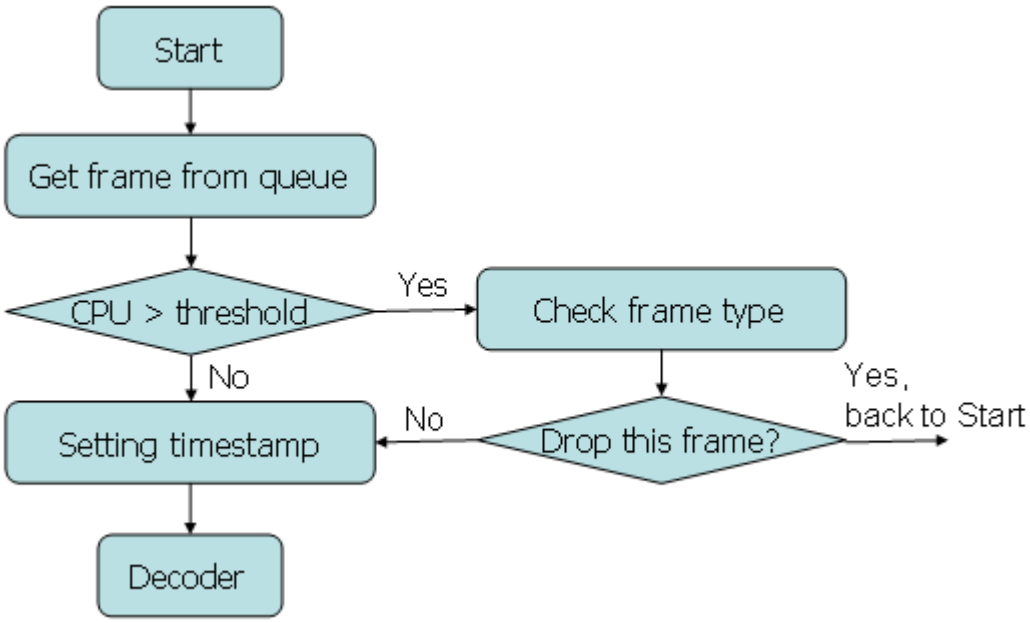



圖 3-2 Frame controller 運作流程圖

當播放器收到一段 video 時，會將已存在 video frame queue 中每個 frame 的資訊依序傳給 frame controller，其中對於 frame controller 有用的資訊包括這個 frame 所代表的種類：I frame、P frame、或是 B frame，以及這個 frame 在原本 video frame queue 中的 decode timestamp。

Frame controller 收到一組 frame 的資訊後，會先向資源監視器查詢目前系統資源的使用情形，以資源監視器所傳回的資訊來當作判定目前播放器狀態調整的一個依據。

接下來判斷所收到的這個 frame 對於現在這個系統狀態下，所應對的處理方式，首先依照 frame 的種類判斷這個 frame 是否要排進新的 video frame queue 中，如果這個 frame 是必須捨棄的話，就繼續從原始的 video frame queue 取得下一個 frame 的資訊進行判斷，或者決定將這個 frame 放進新的 video frame queue 中。frame controller 還必須照目前系統的狀態來判斷這個 frame 應該在什麼時候進行解碼，進而設定這個 frame 的 decode timestamp。

新建立的 video frame queue 會與原本播放器運作的方式相同，將 packet 依序傳給解碼器及處理顯示部分的函式來進行顯示的工作，完成影片的播放。

## 3.2 資源監視器

資源監視器負責提供關於系統資源使用的資訊，系統資源種類繁多：CPU、記憶體使用量、I / O 工作量、網路負載等等，在資源監視器中實作對 CPU 及記憶體使用量進行監控，但是 CPU 的使用量及影響系統穩定的程度較記憶體上爲明顯，所以在本節後只對 CPU 的使用情形進行討論。

在 Linux 架構下，關於系統資源的資訊及程序運作情形都放在根目錄下 proc 的資料夾中，資源監視器必須先從儲存資源使用的檔案中解析出關於 CPU 及記憶體的資訊，接下來透過計算之後獲得目前系統中 CPU 與記憶體的使用量。下面所列出的是 CPU 與記憶體相關資訊所儲存的位置：

- CPU 資訊：/proc/stat
- 記憶體資訊：/proc/meminfo

資源監視器提供使用者對監控的資源進行上限值的設定，若發現所監控的資

源使用超過上限值時傳遞一個訊息給 frame controller 進行播放器模式的調整。

# 3.3 Frame controller

代表一段影片的 video frame queue 中，由三種不同的 frame 所組成，分別爲：I frame、P frame 以及 B frame。原始的影片播放方式就是照著序列中 frame 的順序以及 frame 裡面所記錄的 decode timestamp 順序進行 decoder。

Frame controller 的主要工作有兩個：第一個是依照系統資源監視器所提供的資訊進行播放器模式的調整，第二個就是要在原本的 video frame queue 中，依照目前播放器的模式，挑選出要排進新序列的 frame，同時對排進新序列的 frame 所代表的 decode timestamp 進行調整。

## 3.3.1 基本規則

在本篇論文我們先設定播放器共有兩種模式可供調整，分別是一般模式以及低運算量模式。對於兩種不同的模式，我們對兩個模式中不同種類的 frame 設定了最基本的規則來作爲放進新的 video frame queue 之依據。

- 播放器處於一般模式：

  一般模式下對於每個 frame 有兩個共通的基本規則：

  1. 在一般模式下不進行捨棄 frame 的動作
  2. Frame 與 frame 之間的時間間隔與原始 video frame queue 中相同

- 播放器處於低運算量模式：

當播放器調整至低運算量模式時，對於每種 frame 有不同的規則：

1. I frame：I frame 無論在提供資訊以及解碼程序上都是重要的依據，所以即便在播放器處於低運算量模式時，對於 I frame 與在一般模式時的作法相同，以原始 I frame 所設定的 decode timestamp 作爲依據，當作置入新的 video frame queue 時所存入的 decode timestamp，另外 I frame 一定會存在新的 video frame queue 中而不被捨棄。

2. P frame：對於 P frame 而言，當播放器處於低運算量模式進行播放時，如果有需要 frame controller 會將這個 P frame 作捨棄的動作，如果決定這個 frame 要放置到新的序列中，則 frame controller 會照著設定好的時間間隔來設定此 P frame 的 decode timestamp。

3. B frame：在低運算量模式下，frame controller 對於 B frame 的處理爲直接將該 frame 捨棄，也就是說若處於低運算量模式時，這個 frame 不會出現在新的 video frame queue 中。

### 3.3.2 播放模式介紹

在本小節中會以圖示的方式來解說在兩個不同的播放模式下，frame controller 進行 frame 挑選以及 decode timestamp 的設定原理，以完成新的 video frame queue。

- 播放器處於一般模式時：依循前小節所設定的規則，在建立新的序列時，frame controller不會對任何一個frame進行捨棄的動作。而frame與frame的時間間隔也以原本video frame queue中兩個frame的間隔進行設定（如圖 3-3）。

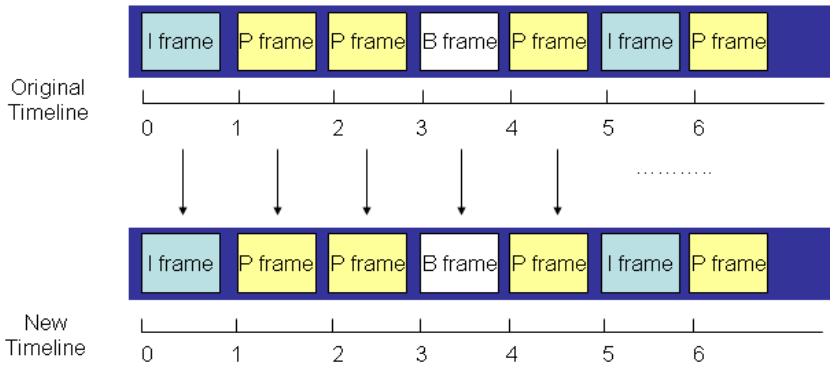



圖 3-3　　一般模式下播放情形

- 播放器處於低運算模式播放時：在低運算模式下，先依照 frame 的種類來作取捨，然後再進行 decode timestamp 的設定，在我們設計的架構下我們打算降低原本影片的 frame rate，所以在低運算模式中 frame 與 frame 之間 timestamp 的間隔設定爲 2。

由圖 3-3中的影片作爲一個範例，可以看到第一個所收到的frame爲I frame，I frame對於整個影片來講是重要的，所以我們對I frame的處理方式是將其排進新的video frame queue中，且維持原本video frame queue所設定好的decode timestamp。第二個frame是P frame，與前一個frame的間隔要將原本video frame queue中的 1 調整成 2，所以這個P frame在新的video frame queue中所設定的decode timestamp將會成爲 2，接下來的frame依然是一個P frame，所以就照上一個frame的作法將這個frame的decode timestamp向後移動到 4 的位置（圖 3-5）。

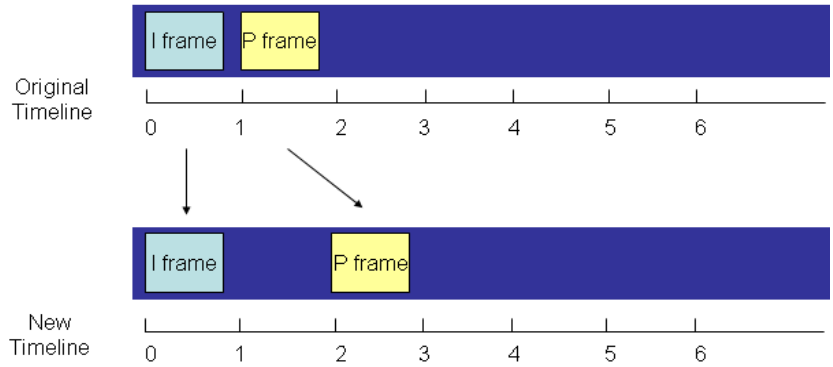



圖 3-4　　1st and 2nd frames : I frame and P frame

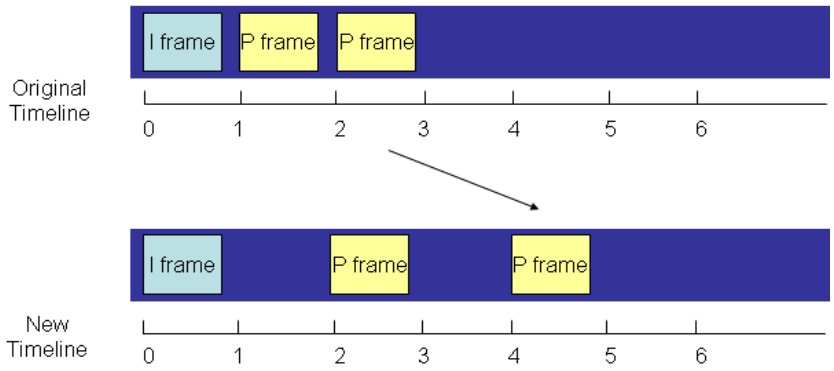



圖 3-5　　3rd frame: P frame

下一個frame是B frame，以前小節基本規則中的設定，frame controller會將B frame進行一個捨棄的動作，所以在新的video frame queue中將不會存在有這個B frame（圖 3-6）。

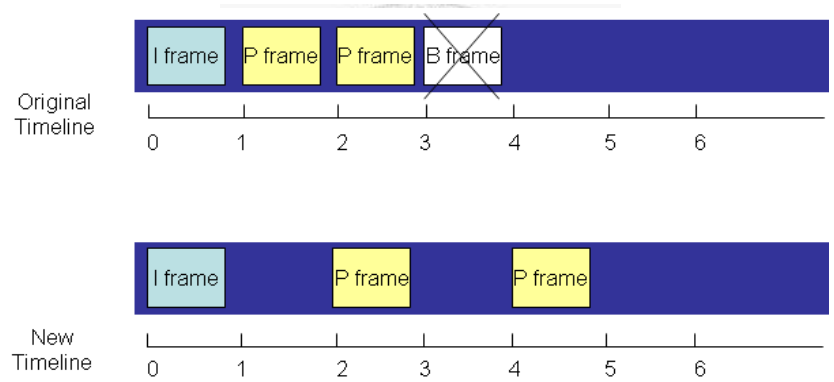



圖 3-6　　4th frame: B frame 之處理

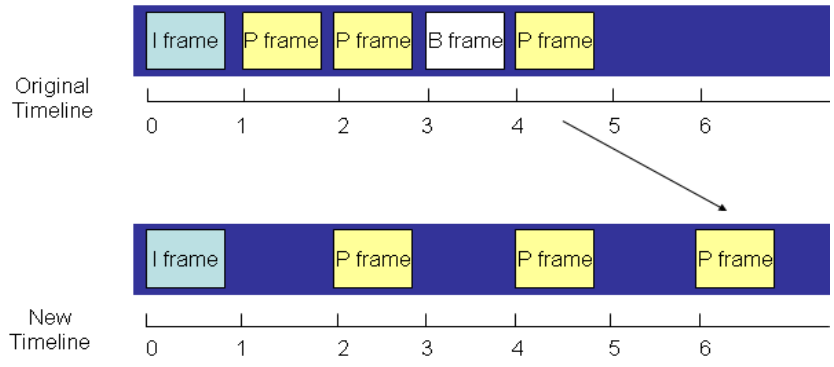



圖 3-7　　5th frame: P frame

若frame controller在處理I frame時，在新的video frame queue中若出現原本順序在此I frame前，但是decode timestamp卻比這個I frame後面的情況時（如圖3-8），我們的作法是將在新的video frame queue中，在I frame decoder timestamp

之後的所有frame進行捨棄的動作，如此一來可以保持整個影片播放時所提供的資訊正確性，同時在解碼上也不會出現錯誤。其餘情況frame controller將繼續以相同的方式進行frame的挑選以及decode timestamp的設定（圖 3-9）。

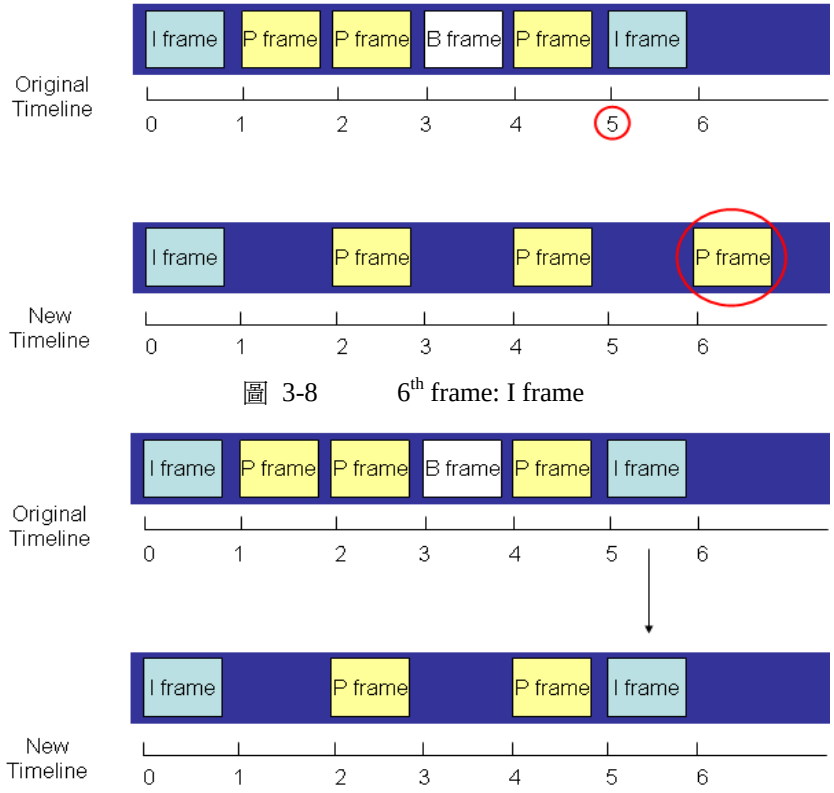



圖 3-8　　6th frame: I frame

圖 3-9　　將 I frame 時間點之後的 frame 移除

## 3.3.3 模式轉換

本小節中將以圖例介紹當模式轉換發生時，frame controller 是如何完成 frame 挑選及 decode timestamp 的設定。

- 由一般模式轉換至低運算量模式

播放器從一般模式轉換至低運算模式較爲簡單，由於一般模式的播放就與原始影片所設定播放的情形相同所以在進行模式轉換的時候不需要注意轉換前處理的 frame 是 I frame、P frame 或者是 B frame。

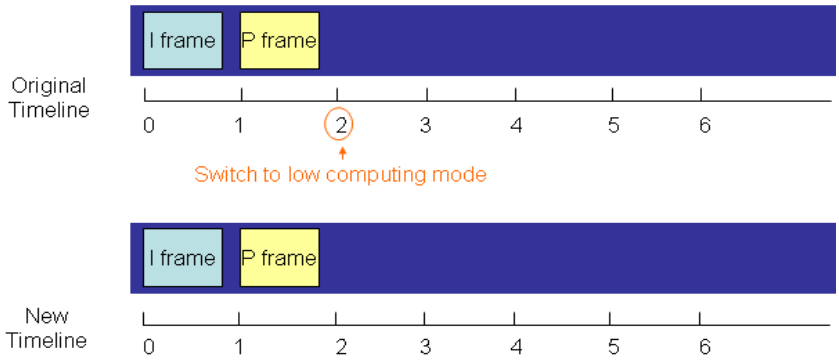



圖 3-10　轉換模式前依照一般模式播放

進入低運算模式之後，我們所設定的frame rate爲原始影片的一半，frame controller在設定decode timestamp時要注意frame之間的間隔要比原始影片多一倍，如圖 3-10所示，原本frame之間的間隔爲 1，進入低運算模式後對於還沒處理的frame就以間隔爲 2 來作設定，然後依序取出在原始video frame queue的frame來進行取捨及設定（圖 3-11）。

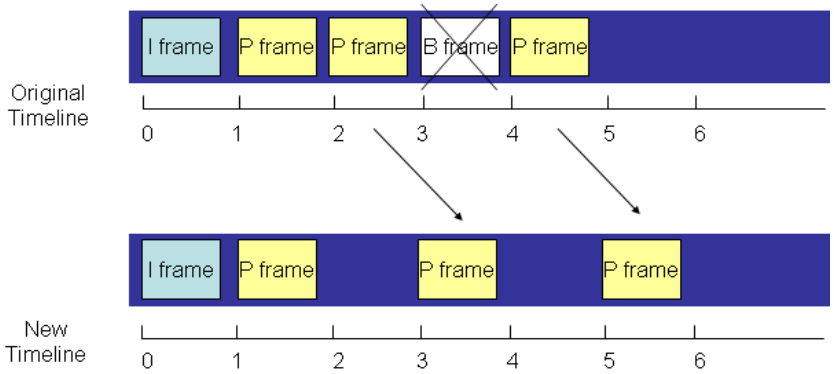



圖 3-11　轉換模式後依照低運算模式播放

其中較需要注意的地方爲前小節所提到在低運算量模式下對於I frame需要注意，也就是當下一個frame爲I frame且在新的video frame queue中已經存在了在此I frame decode timestamp之前的frame，frame controller需要將這個decode timestamp之後作一個清空的動作，以維持影片播放的正確性（圖 3-12）。

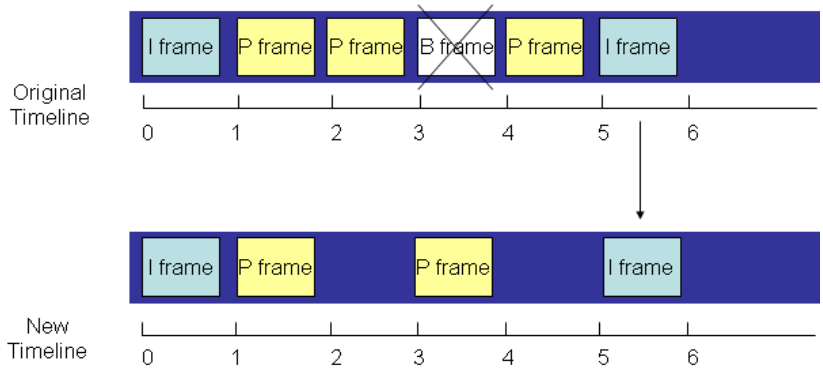



圖 3-12　處理 I frame decode timestamp

- 由低運算量模式轉換至一般模式

從低運算量模式轉換回一般模式時，由於在低運算量模式下，我們會對一些 frame 進行捨棄的動作，所以在切換回一般模式時，必須一些不同的情況做出不同的應對方式，提供解碼器正確且可解碼的 video frame queue。

1. 轉換模式前爲 I frame：若轉換模式前爲 I frame，frame controller 不用對接下來的 frame 進行挑選的工作，就與一般開始播放影片的動作相同。

2. 轉換模式前爲P frame：在P frame之後若下一個frame是I frame時，由於此I frame前都是以低運算量模式作播放，frame間隔變成原始影片的兩倍（圖 3-13），若在新的video frame queue中有frame的decode timestamp超過這個I frame，就依照前小節所敘述之判斷方法來執行（圖 3-14、圖 3-15）。

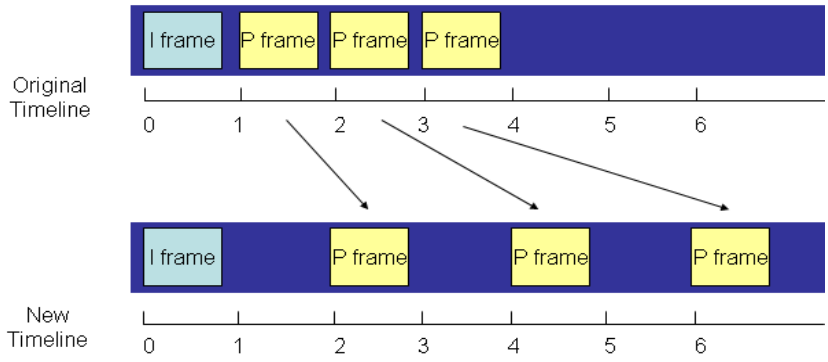



圖 3-13　依照低運算模式規則

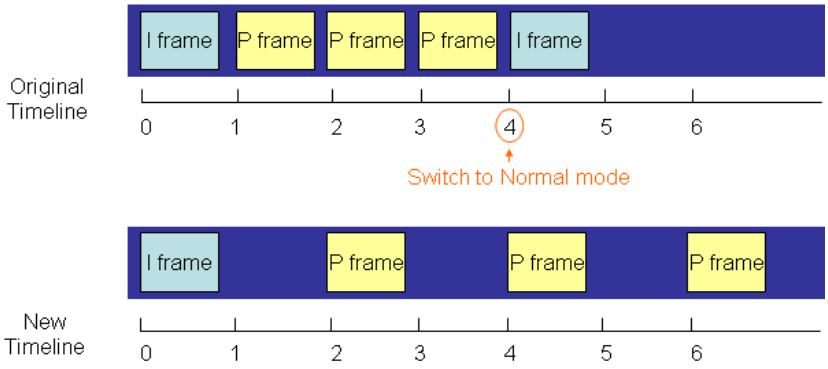



圖 3-14　切換至一般模式時下一個 frame 爲 I frame

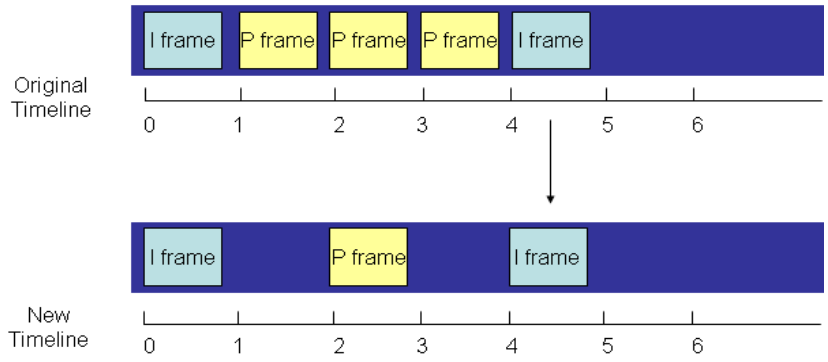



圖 3-15　依照對 I frame 的處理方式將比 I frame timestamp 後的 frame 清空

若P frame之後是接著P frame或者是B frame的情況（圖 3-16），則將此接收到的frame以前一個frame的decode timestamp + 1 來作設定（圖 3-17）。

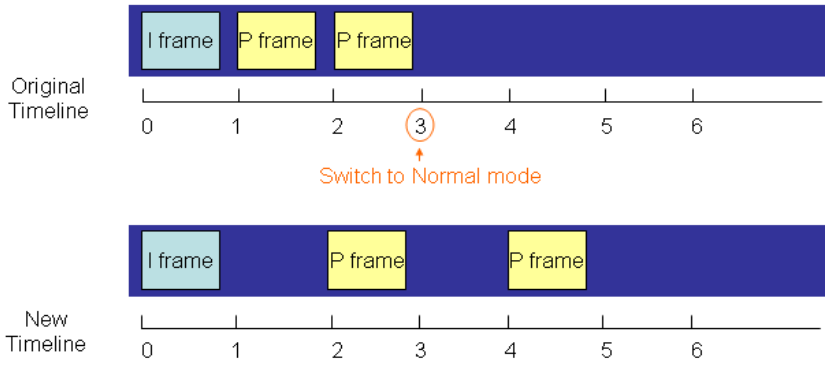



圖 3-16　P frame 之後進行模式轉換

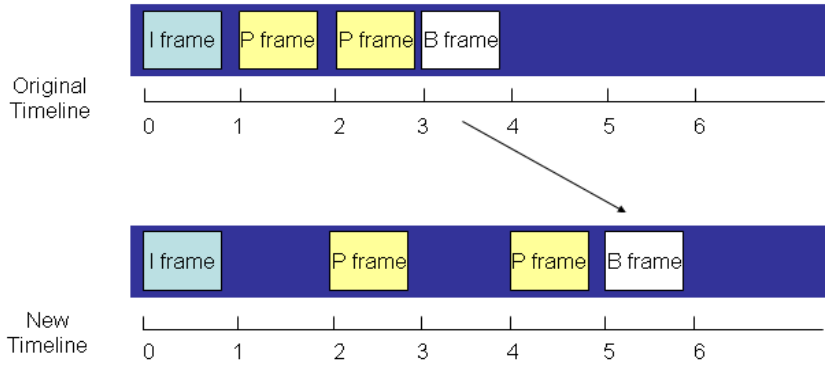



圖 3-17　轉換後下一個 frame 爲 B / P frame 的處理方式

3. 轉換模式前爲B frame：在低運算量模式下，此B frame是會frame controller被捨棄的。此時將會有兩種情形發生：若存在新的video frame queue中最後一個frame的decode timestamp比這個B frame的decode timestamp小（圖 3-18），模式轉換後將會由於轉換前的這個B frame被捨棄而造成轉換模式前新的video frame queue中最後的一個frame播放時間延長出現延遲的情形（圖 3-19）；

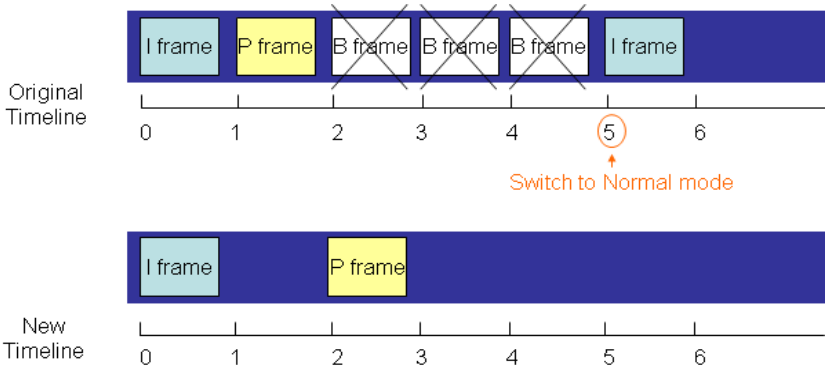



圖 3-18　在 B frame 之後轉換模式的下一個 frame 爲 I frame

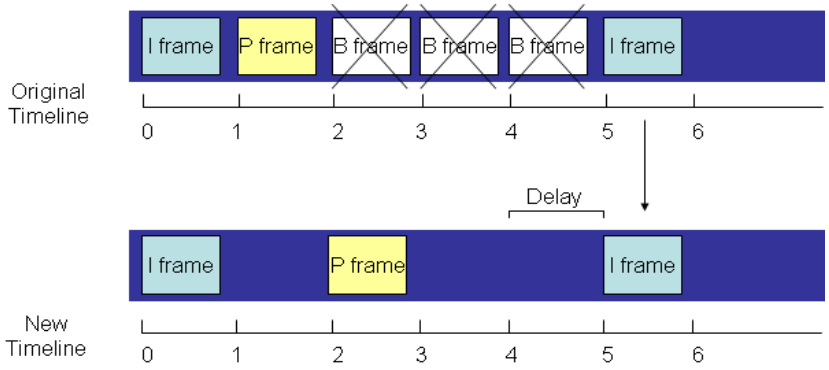



圖 3-19　前一個 P frame 顯示時間延長

另一種情形則是轉換前的B frame的decode timestamp較小（圖 3-20），則

下一個收到的frame若爲P或B frame（圖 3-21），會被排到新的video frame queue的最尾端，decode timestamp將設定爲前一個frame的decode timestamp + 1；若轉換之後下一個frame爲I frame則依照前述方式檢查新的video frame queue及設定decode timestamp（圖 3-22）。

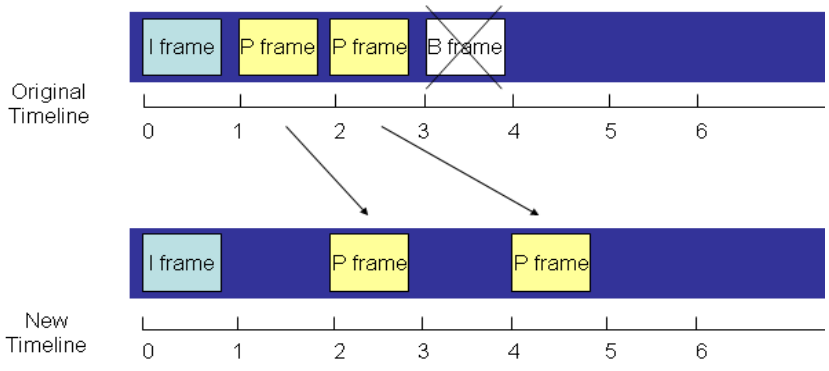



圖 3-20　轉換成一般模式之前

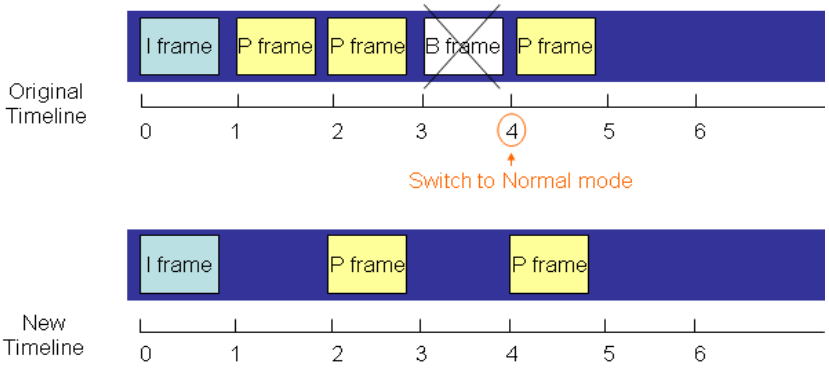



圖 3-21　轉換模式後下一個 frame 爲 P frame

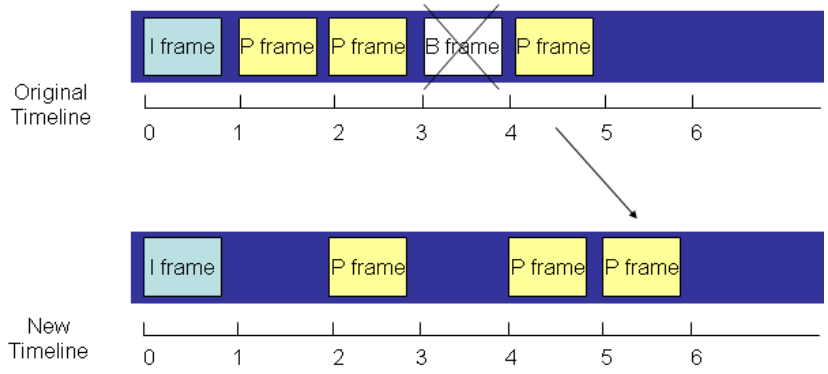



圖 3-22　轉換後的下一個 P frame 的位置

# Chapter4.系統效能評估

## 4.1. 測試環境及方法

爲了了解系統的效能，我們做了一個記錄 CPU 使用量的記錄程式以測試播放器在不同模式下 CPU 的使用情形。測試的環境如下：

| | CPU | Memory | OS |
|---|---|---|---|
| Linux system | Intel P4 3.20GHz | 2G | Ubuntu 7.04 |

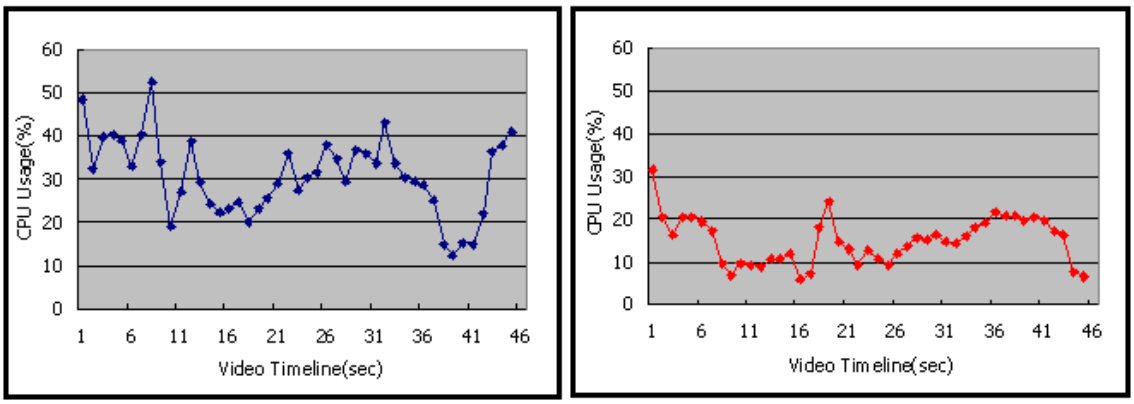
圖 4-1　效能測試環境

我們將播放器與記錄程式同時開始，記錄程式每秒會記錄一次 CPU 的使用量，我們將對兩個內容不同影片，其格式皆爲 H.264、24fps、原始長寬爲 1280*720 的影片，在我們所設計的播放器下，以 640*480 的大小分別進行三個不同的測試：一般模式播放、低運算量模式播放以及兩種模式交互切換，其中模式交互切換的時間點爲每 300 的 frame 進行一次模式的切換。

## 4.2. 測試結果與說明

透過記錄程式，將三種不同的測試結果製作成折線圖表示 CPU 的使用情形：

- 影片 1：




(a)　(b)

圖 4-2　影片 1 於 (a)一般模式播放 (b)低運算量模式播放 CPU 使用情形

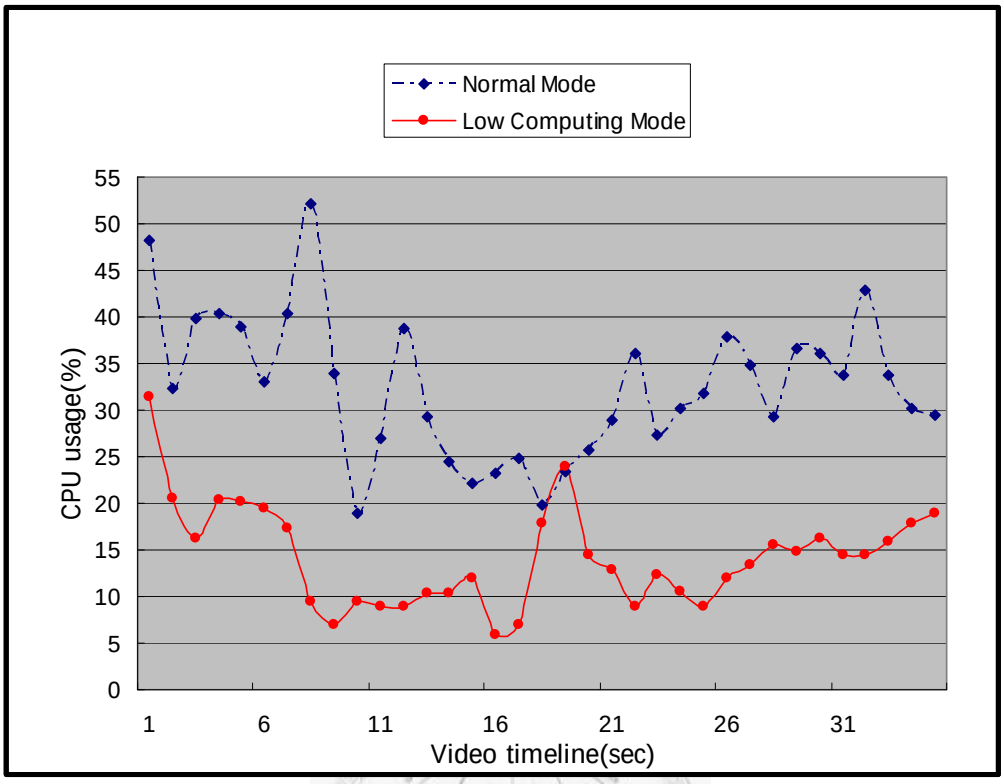



圖 4-3　　影片 1 一般模式與低運算模式 CPU 使用量比較圖

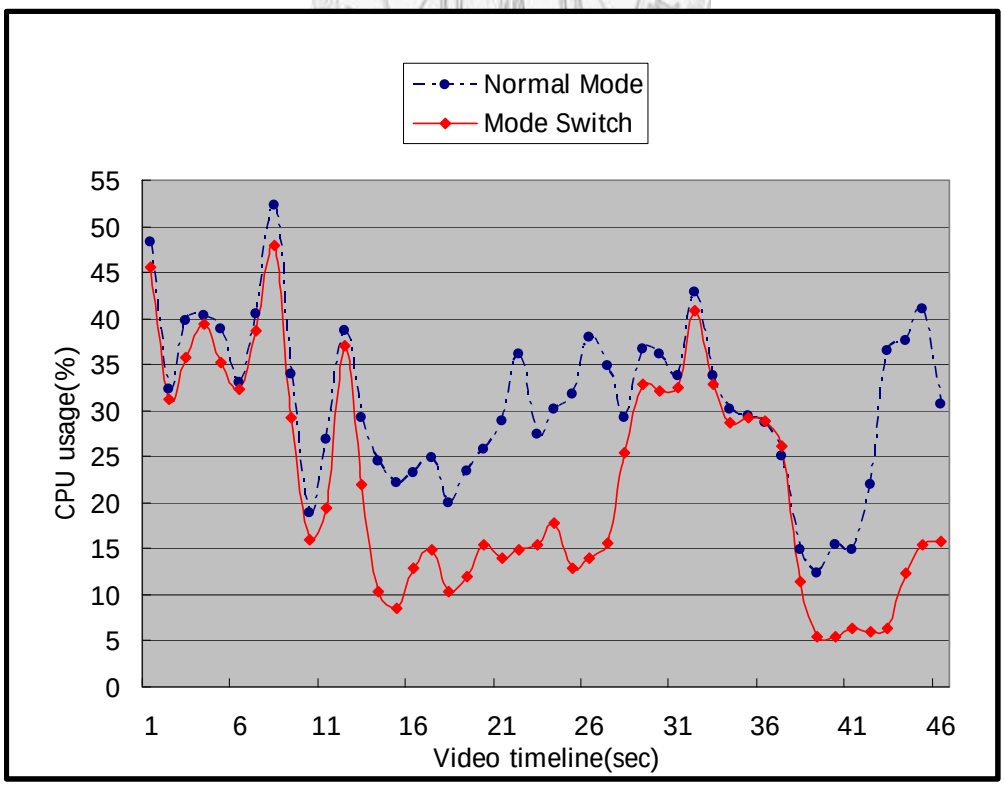
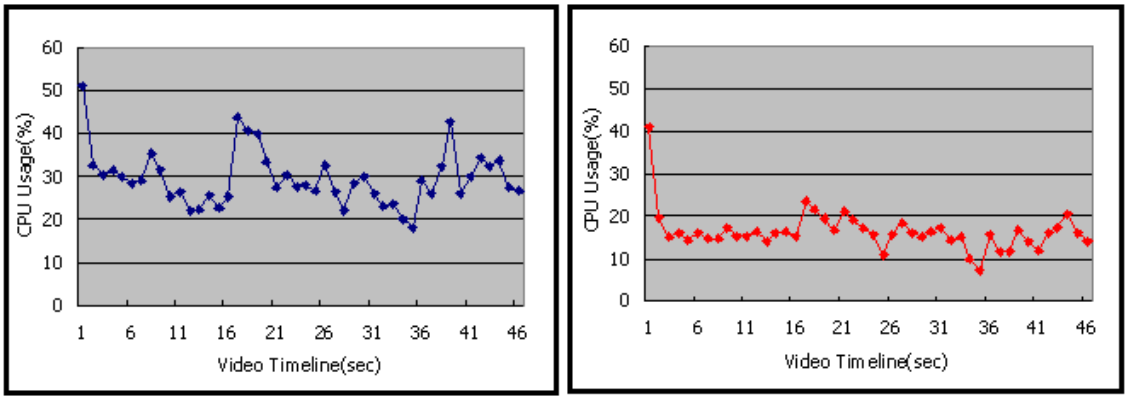



圖 4-4　　影片 1 一般模式與兩種模式交互切換 CPU 使用量比較圖

- 影片 2




(a) (b)

圖 4-5 影片 2 於 (a)一般模式播放 (b)低運算量模式播放 CPU 使用情形

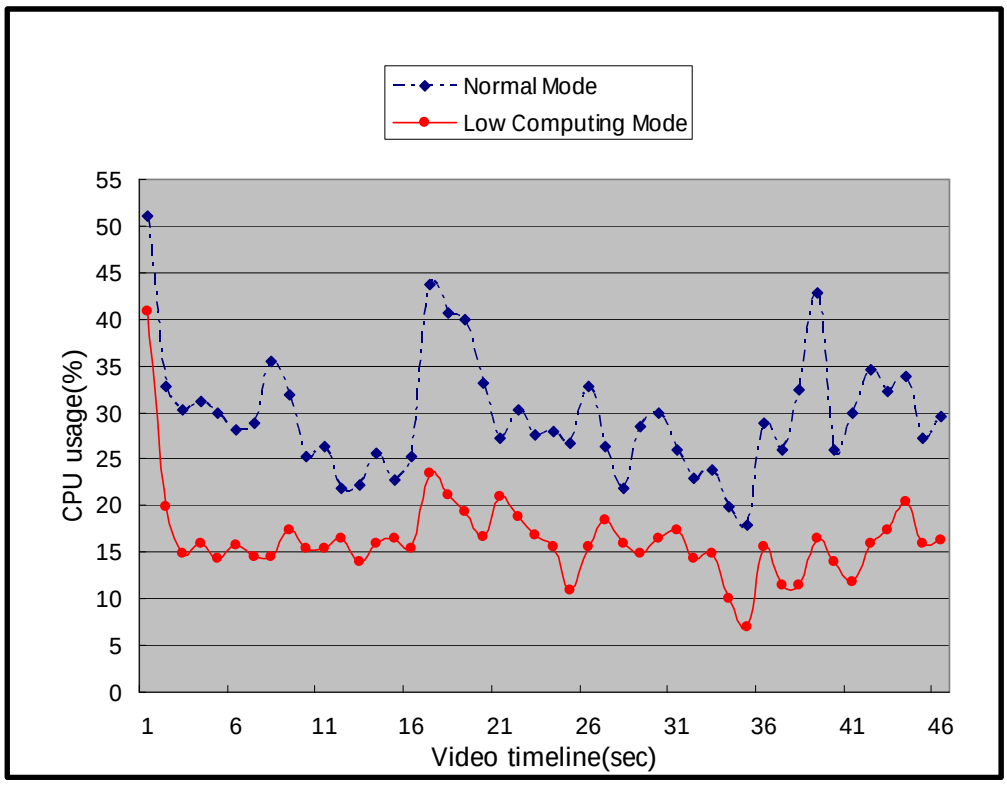



圖 4-6 影片 2 一般模式與低運算模式 CPU 使用量比較圖

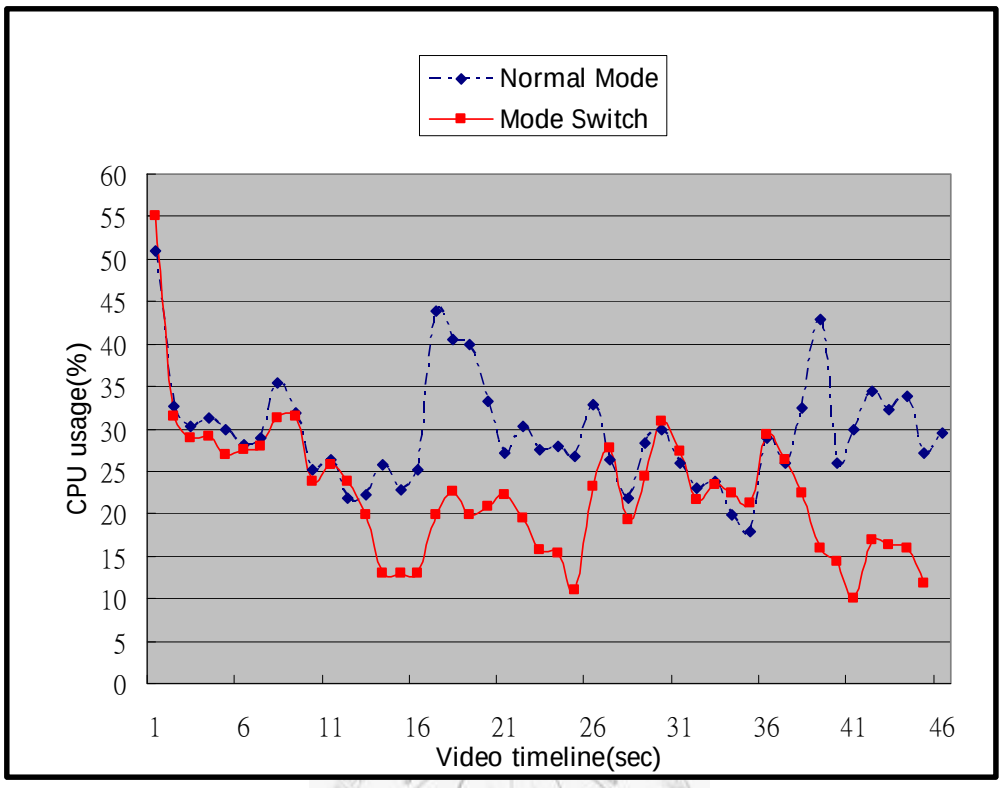



圖 4-7　影片 2 一般模式與兩種模式交互切換 CPU 使用量比較圖

由以上的曲線圖中我們可以發現當播放器轉換成低運算量模式時，CPU的使用量都有明顯的降低，從圖 4-8中可以發現：當影片 1 以低運算量模式播放時，較一般模式減少 15.88%的CPU使用量；而影片 2 以低運算量模式播放時，也降低了 13.20%的CPU使用量。所以在低運算量模式下，frame controller能夠有效降低播放器的CPU使用量。

| | 一般模式下 CPU 平均使用量(%) | 低運算量模式下 CPU 平均使用量(%) | 減少 CPU 使用 (%) |
|---|---|---|---|
| Video 1 | 30.756 | 14.876 | 15.88 |
| Video 2 | 29.551 | 16.353 | 13.198 |

圖 4-8　一般模式與低運量模式 CPU 使用量比較表

# Chapter5.結論與未來工作

## 5.1. 結論

系統穩定性對於系統及程式設計上十分重要，當系統發生運算量過大系統無法負荷、資源分配不均或是資源不足都會導致系統不穩定。尤其是嵌入式系統以及手持裝置日漸普及，在這些平台上所要完成的工作及提供的服務也越來越多且與一般電腦系統越來越相似。嵌入式系統與手持裝置上所能提供資源以及運算能力和一般電腦或系統比較起來是少了許多。所以更容易在系統運作時遇到資源分配不均或者是運算能力不足而導致整個系統在一個不穩定的狀態下運作。

本篇論文提出一個 MPEG-4 的多媒體播放架構，此架構能使系統資源使用量超過可負載的上限值時，藉由改變播放器的模式，調整其運算量以提供整個系統回復到穩定狀態。

播放器模式的改變主要由 frame controller 依照系統資源監視器所提供的資訊進行調整，只要有資源使用上的變化，播放器可以依照資源使用情形來降低播放器所需要的運算量。

Frame controller 對於降低播放器的運算量扮演重要的角色，對於每個 frame 依照不同的種類及所在的模式進行新序列的建立，使得新的序列中所存在的 frame 個數比原始 video frame queue 中的個數少，這麼一來需要作解碼工作的 frame 數量降低，運算量也跟著減少。在低運算模式下，挑選 frame 的演算法會讓一些 frame 從原本的 video frame queue 到新的序列時被捨棄而產生一些影片中資訊些微的損失。

擴大序列中 frame 的時間間隔將會造成延遲的現象，依照該 frame 跟前一個最近的 I frame 和下一個 I frame 的時間點有相當的關係，其中在這裡提出在兩個 I frame 之間所可能產生的最大延遲時間：

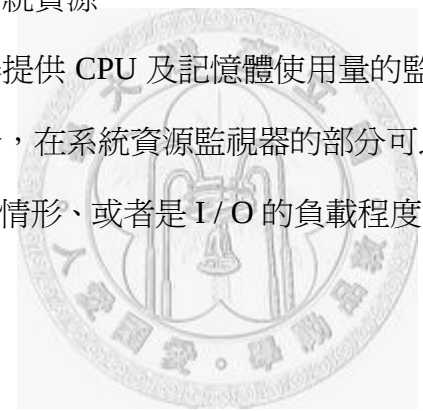
- 在低運算模式下，影片的 frame rate 若爲：f (fps)
- 兩個 I frame 之間的間隔爲：n (秒)

則可以計算出在兩個 I frame 之中的最大延遲時間爲： n / 2f (秒)

## 5.2. 未來工作

- 能監控更多種類的系統資源

目前系統資源監視器提供 CPU 及記憶體使用量的監控，而 CPU 與記憶體只是整個系統資源的一部份，在系統資源監視器的部分可以提供更多種系統資源監控的選擇，如網路的使用情形、或者是 I / O 的負載程度等等，都可以加入系統資源監視器中進行監控。



- 更多不同的播放器模式及調整模式的規則可供選擇

目前架構下的 frame controller 對於系統資源使用的判定，只限於設定一種資源的使用上限值，無法提供多種模式和調整模式的規則供使用者使用。若能讓使用者在對多個不同系統資源進行監控時，分別對不同資源過度使用的情況，產生各自對應的轉換，播放器將可對不同種類的資源提供降低運算量或是減少資源使用的效果。

- 改善序列建立方式以達到減低影片資訊損失且能使延遲情況減低的效果

本篇論文在低運算量模式播放下所用來建立新的序列之方法會使部分的 frame 被 frame controller 作捨棄的動作，造成影片中資訊損失，同時也產生影片

延遲的情形，未來可以改善序列的建立方式，像是平均的在兩個 I frame 之間挑選所要留下來的 frame，這樣將提供使用者一個更好的播放環境，讓影片提供的資訊能夠完整且較精準的傳達給使用者。

# 參考資料


1. ISO/IEC 14496 AM 1. Part 1:System, international organization for standard, 2001
2. Yu-Chen Liu, “Implementation of a CPU Resource Management for MPEG-4 Applications” Master Thesis, 2005
3. Kou-Shin Yang, “Design and Implementation of an MPEG-4 Application Engine,” Master Thesis, 2004
4. Yi-Chin Huang et al, “Design and Implementation of an Efficient MPEG-4 Interactive Terminal on Embedded Devices,” ICME2004, P.715-718
5. “FFMpeg：Multimedia solution with recording, converting, and streaming.” from http://ffmpeg.mplayerhq.hu/
6. David K. Y. Yau and Simon S. Lam, “Adaptive rate-controlled scheduling for multimedia applications,” ACM Multimedia Conference, 1996, P.129-140
7. Klara Nahrstedt, Hao-Hua Chu, and Srinivas Narayan, “QoS-Aware Resource Management for Distributed Multimedia Application,“ Journal of High Speed Networks, Special Issue on Multimedia Networking, 1999, Volume 7(3,4): P.229-258