**Novel Chroma Subsampling Strategy based on Mathematical Optimization for Compressing Mosaic Videos with Arbitrary RGB Color Filter Arrays using Parallel Programming**

簡育聲

國立交通大學

資訊科學與工程研究所

0756056

arashi30801@gmail.com

游依杰

國立交通大學

多媒體工程研究所

0756613

jay60203@gmail.com

劉承順

國立交通大學

資訊科學與工程研究所

0656124

ciaskbe@gmail.com

**ABSTRACT**

這篇論文實驗了使用GPU的平行處理在影像處理的領域的效果。由於對GPU程式的編程介面的優勢，我們主要探討CUDA作為平行處理平台的效果。

根據文獻的彩度採樣流程[1]被我們用於實驗，我們探討使用CUDA在影像處理可能獲得的加速倍率，我們比較了平行處理前後的程式效率。

**CCS Concepts**

• **Information systems➝Multimedia information systems.**

**• Computing methodologies ➝ Massively parallel and high-performance simulations.**

**Keywords**

Image processing; chroma subsampling; H.264/AVC; High Efficiency Video Coding (HEVC); mosaic videos; Parallel Programming; CUDA

1. **Introduction**

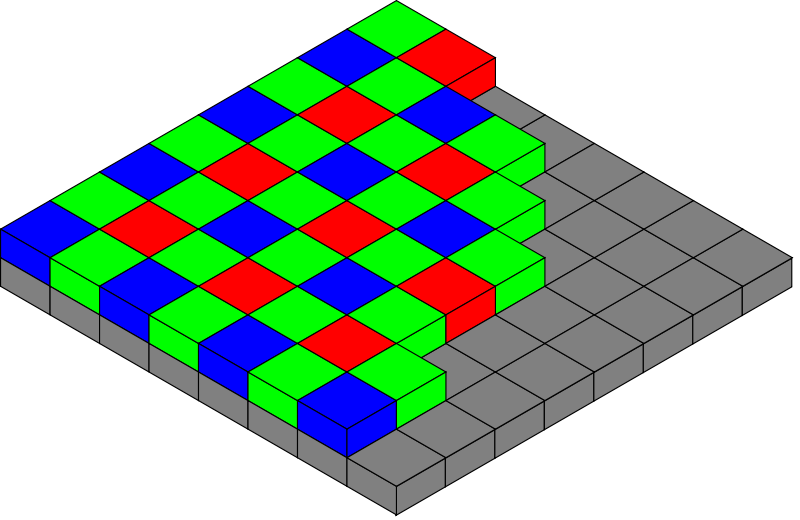
時至今日，為了減少製造成本，幾乎所有數位相機、錄影器、掃描器仍舊使用單傳感器(sensor)，搭載紅藍綠三色濾光片陣列(red–green– blue (RGB) color filter array)來捕捉影像。如此在這種架構下產生的影像被稱之為馬賽克影像(mosaic videos)。

在Ch Lin et al.的論文[1](以下簡稱論文[1])中提出一種彩度採樣流程(Chroma Subsampling) 技術，對每個被依4:2:0 format採樣的2 × 2 YUV區塊而言，其U、V是依照最小化品質失真來決定的。品質失真即是將原本的馬賽克區塊以及YUV區塊還原後的區塊比較。與Chen et al.’s[9]、Yang et al.’s[10]的論文的彩度採樣流程偏好紅、藍(R、B)的重組，論文[1] 的彩度採樣流程可以從其數學最佳化的公式中看出，不特別偏好紅、綠、藍 (R、G、B)。並且，論文[1] 的彩度採樣流程在品質與運算時間的平均表現上較Chen et al. and Yang et al.的方法好。

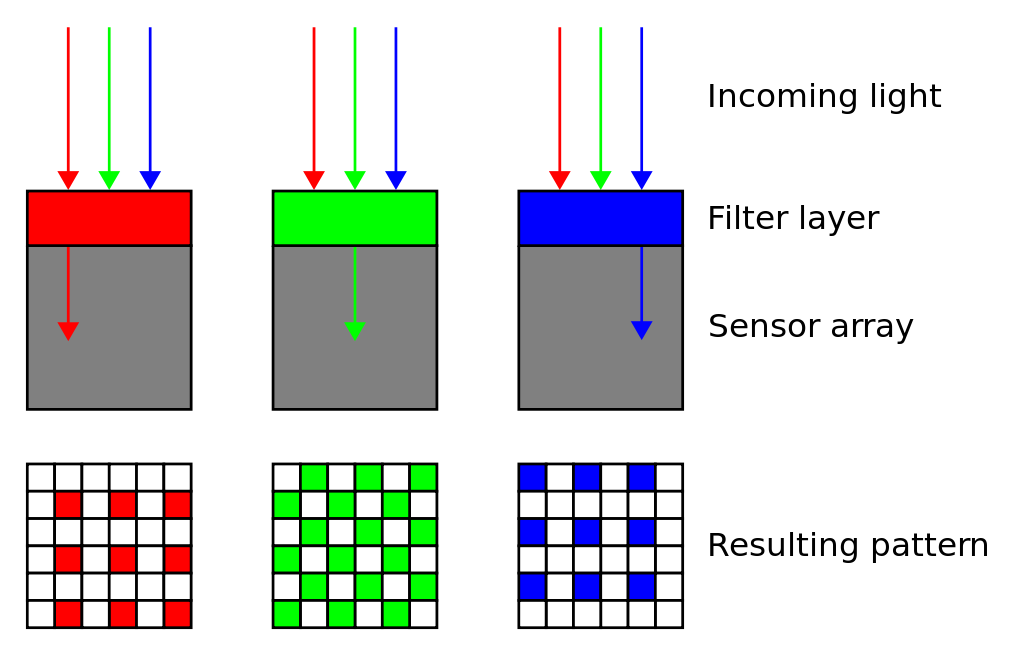
我們選擇這個研究主題的原因是：一、現今的電腦或終端大多都搭載多核心處理器(multi-core CPU)以及顯示卡(GPU)，為了充分利用這些運算資源的優勢。二、考慮到在影像處理的領域主要皆是進行矩陣的運算，並且會耗費相當可觀的運算時間。所以這個研究我們使用平行運算(Parallel Programming)以及GPGPU(General-purpose computing on graphics processing units)的技術來完成壓縮馬賽克影像。將影像處理的工作平行化運算的研究目前並不多，仍有很大的發展空間。本篇主要探討針對論文[1]中出現的矩陣運算進行平行化，包含YUV與RGB間的轉換，以及Chroma downsampling、Chroma upsampling的處理。在影像處理的領域，平行化的加速倍數參考相關的研究結果，可以預估是很高的，可以顯著的減少運算時間。

我們打算實作出論文[1]之文獻內容，並利用107年上學期交通大學平行程式設計課程所學之平行運算對其進行加速運算。

1. **Background Knowledges**
2. **濾光片陣列**

傳感器搭載濾光片陣列(color filter array)來捕捉影像。濾光片陣列有七種主要的種類，不同種類的差異主要在於結構由不同基礎顏色的不同排列組合。濾光片陣列上的各基礎顏色區塊可以透光與其相同顏色的光線。因此，根據傳感器的濾光片陣列排列，會產生相同排列的影像。例如，搭載紅藍綠三色濾光片陣列的話，數位相機的原始影像上的每個像素(Pixel)都只有紅藍綠三色其中的一種，整個影像看起來像是馬賽克紋路一樣。所以一般都需要一個演算法把像素上的其他顏色"推算"出來，以變成一張正常的影像，這個演算法通常也叫去馬賽克演算法。

▲ 圖1：濾光片陣列示意圖，摘自維基百科(https://zh.wikipedia.org/wiki/拜爾濾色鏡)



▲ 圖2：感應器的橫截面，摘自維基百科(https://zh.wikipedia.org/wiki/拜爾濾色鏡)

對一個馬賽克影像來說，其RGB-CFA structure的資訊相當重要，因為將這個資訊進行去馬賽克運算[2] (demosaicking process)，即可將馬賽克影像轉換為全彩影像(fullcolor video)。RGB-CFA structure有7種不同種類，其中又以拜爾濾色鏡(Bayer CFA) 種類為主流。

拜爾濾色鏡的排列有50%是綠色，25%是紅色，另外25%是藍色，因為這樣最接近[人眼](https://zh.wikipedia.org/wiki/%E4%BA%BA%E7%9C%BC)的生理性質，也被稱做RGBG ，GRGB，或者RGGB。

1. **去馬賽克演算法**

去馬賽克有不同的實現方法。一個普遍的演算法是對相鄰同色的像素數值進行內插。舉例來說，當晶片曝光得到一張影像後，每個像素就被確定。濾光片陣列的綠色區塊生成的像素精確測量了綠色成分，而該像素紅色和藍色的成分則是從鄰近像素取得。一個綠色像素的紅色數值可由相鄰兩個紅色像素內插計算出來；同樣的，內插相鄰兩個藍色像素也能計算出藍色數值。

內插法在顏色恆定或均勻變化時表現良好，例如在自然景觀照的狀況。但在顏色和亮度突變處卻會產生雜訊，比如滲色（Color bleeding），在銳利的邊角處特別明顯，會使照片看起來模糊，例如文字影像。因此，其它更精密的去馬賽克的方法嘗試辨認高對比的邊緣，然後僅僅順著這些邊緣做內插，而不越過邊緣。

從[數位相機](https://zh.wikipedia.org/wiki/%E6%95%B8%E4%BD%8D%E7%9B%B8%E6%A9%9F)取出[原始圖檔](https://zh.wikipedia.org/wiki/RAW)後，可以使用專門的電腦軟體來選擇不同的去馬賽克演算法處理，專業的攝影師可以根據原始圖檔的主題、細節和紋理的差異，選擇不同的去馬賽克演算法，由去馬賽克所產生的色彩雜訊也可以成為辨別偽造圖片的重要線索。

1. **H.264 / MPEG4 / AVC 壓縮編碼**

H.264 屬於 MPEG-4 的第10部份，是由ITU-T影像編碼專家協會( VCEG )和ISO/IEC動態圖像專家協會( MPEG )共同組成的聯合影像協會( JVT = Joint Video Team )提出的高壓縮數位影像編解碼器標準。H.264有許多名稱，最常被稱呼的是H.264、AVC(進階影像編碼Advanced Video Coding)。

H.264/AVC專案的目的是為了建立一個更佳的視訊壓縮標準，在更低的位元速率(bitrate)的情況下依然能夠提供良好視訊品質的標準。

1. **CUDA**

CUDA（Compute Unified Device Architecture，統一計算架構）是由[NVIDIA](https://zh.wikipedia.org/wiki/NVIDIA)於2007年所推出的一種整合技術，是該公司對於[GPGPU](https://zh.wikipedia.org/wiki/GPGPU)的正式名稱。使用CUDA可以利用GPU作為C-編譯器的開發環境，CUDA可以相容[OpenCL](https://zh.wikipedia.org/wiki/OpenCL)或者的C-編譯器。

以[NVIDIA](https://zh.wikipedia.org/wiki/NVIDIA) [GeForce 8800 GTX](https://zh.wikipedia.org/wiki/GeForce_8)顯示卡為例，其核心擁有128個內處理器。利用CUDA技術，就可以將那些內處理器串通起來，成為執行緒處理器去解決資料密集的計算，運算能力可達到520GFlops。而各個內處理器能夠交換、同步和共用資料。

程式設計者可以利用CUDA的C語言擴充 (extension) 直接用C語言寫程式，設計資料分配 (data decomposition) 及程式流程將運算工作分配到上千個執行緒(threads)及圖形處理器中數以百計的計算核心 (cores)。CUDA可以運作在NVIDIA GeForce 8系列之後的GPU上，現在常見的一張二千元的NVIDIA顯示卡，就能進行CUDA運算而且效能驚人，所以GPGPU也算是一種平價的高效能運算方式，它把桌上型電腦變成一台個人的超級電腦 (personal supercomputer in a desktop)。

在CUDA程式中，有兩個不同的運算環境：host及device。Host就是原本中央處理器的計算環境，可以讀寫檔案、配置記憶體、使用外部函式庫、呼叫和傳遞參數給GPU的副程式。Device是指GPU，它有獨立的記憶體和計算核心，計算用的資料需要從host傳送到device上的記憶體，才能在device中處理。在device上執行的副程式，稱為kernel，通常有上百到上千個執行緒(thread)執行同一個kernel。

每一個執行緒會屬於一個執行緒區塊 (thread block)，每一個執行緒區塊裡的執行緒總數有上限，不能超過512個執行緒，執行緒區塊可以指定一維到三維的排列方式來排列執行緒。所有的執行緒區塊又會以一維或二維的方式排列在格子(grid)裡，每次Kernel要啟動的時候，就會以kernel\_function<<<blocks, threads>>>(…) 語法，來宣告格子(grid)及執行緒區塊的排列方式。每個thread及block依不同的排列順序會有不同的編號 (thread/block ID)，藉由執行緒編號，資料可以分配到不同執行緒上進行處理。圖一顯示一個blocks為(m, n)、threads為(2, 2)的grid，其中block和thread的排列方式。

CUDA將常用到的數學函數，如sin()，exp()等包裝進編譯環境裡，這些提供的數學函數就可以直接在device程式中呼叫。如果不需要那麼高的精確度，而是需要率效的話，也有圖形處理器內建的數學函數 (Intrinsic function) 可以使用，函數名就是原本的函數前加入底線，如 \_\_sin()、\_\_exp()，這些函數直接對應到GPU上的指令集，所以效率很高。

1. **圖形處理器介紹**

顯示卡的處理器稱為圖形處理器（GPU），與CPU類似，只不過GPU是專為執行複雜的數學和幾何計算而設計的，這些計算是圖形渲染所必需的。GPU已經不再局限於3D圖形處理了，在浮點運算、並行計算等部分計算方面，GPU可以提供數十倍乃至於上百倍於CPU的性能。GPU通用計算方面的標準目前有OpenCL、CUDA、ATI STREAM。

圖形處理器的硬體是由許多Streaming Multiprocessors (SMs) 及global memory組成。每一個SM裡還有數個Scalar Processors (SPs)、shared memory、指令提取分派 (instruction fetch/dispatch)、雙倍精度浮點數 (double precision unit) 等控制器。程式中的一個block會分配到一個SM上面執行，block中的thread會分配到這個SM的SP上執行，因此同一個block中的所有thread都可以看到共同的share memory區段，也可以進行同步指令 (synchronize)。以GeForce 9500GT為例，有4個SM、最高1GB的global memory，每個SM上有8個SP、16KB的share memory。

圖形處理器有獨立的記憶體，在進行計算之前，host端程式需要以cudaMalloc()在device上的global memory配置記憶體，再用cudaMemcpy()搬移記憶體內容，在device上執行的kernel才能取得資料。如果要在global memory及share memory間移動，要在kernel裡搬移。計算結束後，把資料搬回host端，再用cudaFree()釋放記憶體。

CUDA程式跟C程式類似，撰寫完程式碼之後，都要先編譯 (compile) 之後才能執行。但是不太一樣的部份是，CUDA的kernel及device副程式都是在圖形處理器上進行運算，編譯器 (compiler) 會進行比較特殊的處理。一個副檔名是.cu的程式碼，編譯器會分成host及device兩個部份處理。host部份的程式碼就依一般的C程式的編譯方式，產生可執行的機器碼。而device部份的程式碼會由NVIDIA提供的編譯器，產生成PTX (Parallel Thread eXecution) 環境的組合語言 (assembly)，在程式要被送到圖形處理器執行時，才被圖形處理器的驅動程式 (driver) 組譯成不同圖形處理器可以執行的機器碼。這種設計的好處是不用在編譯時就決定圖形處理器核心類型，可以相容不同型號的GPU核心，但也使得device程式沒辦法直接直接呼叫現成的函式庫，目前也還不支援函式遞迴呼叫 (recursive function call)。在使用nvcc編譯時加上-ptx可以產生PTX組合語言碼，加上-cubin可以產生kernel程式使用register、shared memory的數量資訊，這兩種資訊對程式最佳化很有幫助。

圖形處理器的運算能力進步飛快，以大量處理核心加速運算的方式，更是領先傳統的中央處理器，尤其在中央處理器放棄追求高時脈之後、改走多核心提高效能的現在，更是顯示出圖形處理器開發人員的遠見。相信將來，也會有圖形處理器組成的超級電腦，擠身在五百大超級電腦的列表中。

1. **壓縮影像的方式**

近十年來，多個壓縮影像的方式已被提出，主要可以分成兩種：一、壓縮優先。二、去馬賽克運算優先。壓縮優先的方式經由顏色分割、幾何校正先將CFA structure轉換，然後直接進行壓縮，可以獲得較好的壓縮比例，經常應用在Bayer CFA格式，因為Bayer CFA格式每個2 × 2的區塊的綠：藍：紅(G:B:R)的比例為2:1:1，與加密器(encoder)的4:2:2 subsampling format相近。

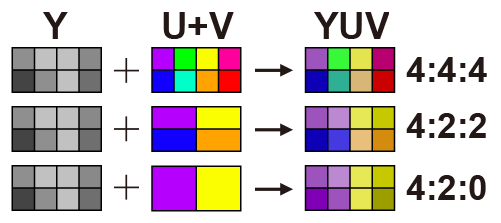
去馬賽克運算優先，首先對馬賽克影像進行去馬賽克運算，然後進行常規的壓縮，包含RGB對YUV之轉換、彩度採樣流程(Chroma Subsampling)、視頻編碼(video coding，如H.264/AVC、HEVC)以使所有RGB-CFA structures皆可使用。一個簡單的去馬賽克運算優先的壓縮，至少採用了一種去馬賽克運算技術，以及一種對全彩影像的彩度採樣流程(Chroma Subsampling) 技術。

隨著影像解析度的不斷提升，H.264/AVC

標準已經無法滿足影像壓縮的要求。因此高效率視訊編碼(HEVC)應運而生，HEVC又稱為H.265和MPEG-H第2部分，是一種[視訊壓縮](https://zh.wikipedia.org/wiki/%E8%A6%96%E8%A8%8A%E5%A3%93%E7%B8%AE)[標準](https://zh.wikipedia.org/wiki/%E6%A0%87%E5%87%86)，被視為是[ITU-T H.264/MPEG-4 AVC](https://zh.wikipedia.org/wiki/H.264/MPEG-4_AVC)標準的繼任者。其相較於H.264/AVC而言，提升了壓縮比例以及解壓縮後的品質，被認為同樣畫面品質下位元率減少到了50%，並且支援的最高解析度可達到8192×4320（8K解析度），僅增加了一些處理時間。

1. **YCbCr 4:2:0**

YCbCr 中，Luminance，指「光度」，即三原色 (紅、綠、藍) 相加；「C」即 Chrominance，指「色度」 ，即是把訊號壓縮成一速訊號來傳送。Cb 即藍與綠的色度，Cr 即紅與綠的色度 是另外兩個不同顏色分量的色度。

由於人眼對亮度特別敏感，反而對色彩份量、對顏色的位置與移動不及亮度敏感，再加上「影片」與「相片」不同，因為影片的影像不停轉換，能夠看到的瑕疵自然相對較少，影片是不停轉換中的影像，「相片」屬於定格影像，比較容易看到瑕疵；所以，工程師便可以為不停轉換中的「影片」設計一套運算方法把光與色拆開而能符合非常有限的頻寬要求。

▲ 圖3：YCbCr 4:4:4、4:2:2 與 4:20 的差別

4:4:4、4:2:2 與 4:20 是 YCbCr 的取樣方法。YCbCr 4:4:4 即是無通道壓縮的全像傳送；YCbCr  4:2:2 即是把後面兩個通道的訊號抽掉一半，所以只需要 YCbCr 4:4:4 的三分之二頻寬；至於 4K UHD BD 所用的 4:2:0 方式更加複雜，被抽走藍與紅的色度更加多。

RGB 與 YCbCr 4:4:4 的數據量與影像素質理論上是畫上等號。在 HDMI 及光碟的頻寛及儲存量限制下，則可以用 YCbCr 4:2:2/4:2:0 取樣方式來減低對頻寬或儲存量的要求。若然片源是完整的檔案，RGB 與  YCbCr 4:4:4 格式理論上應該沒有分別。

1. **Statement of the problem**
2. **壓縮影像的重要性**
   1. **儲存容量與傳輸速率的限制**

當畫面被相機截取時，為了降低製作成本，第一手儲存照片的裝置如相機、手機的儲存空間十分有限。近年來快速普及的手機照相功能隨著畫素增長與影像品質增加，檔案大小也越來越大。若儲存在本地端，手機儲存容量對客戶來說是極為重要的考量；若要分享照片或儲存在雲端硬碟，傳輸照片時的網路傳輸速量也是收費的，因此壓縮影像十分必要，並具有實際效益。

* 1. **影像資訊與日俱增**

近年來，行動裝置（Mobile device）蓬勃發展，包括[手機](https://zh.wikipedia.org/wiki/%E6%89%8B%E6%9C%BA)、[筆記型電腦](https://zh.wikipedia.org/wiki/%E7%AC%94%E8%AE%B0%E6%9C%AC%E7%94%B5%E8%84%91)、[平板電腦](https://zh.wikipedia.org/wiki/%E5%B9%B3%E6%9D%BF%E9%9B%BB%E8%85%A6)，甚至是車輛的行車紀錄器或是物聯網(IoT)等，公共設施如監視鏡頭與日俱增。影像與視頻的拍攝急速增長，應用層面廣泛。

在 2016 年時全球總資料大小加起來達到 Zettabyte(byte)，其中近四分之一是照片、電話等原創資料。2015年，[谷歌](https://www.iwiki01.com/3853ed273b89687a)的YouTube有超過10億用戶，世界上所有上網的人群中幾乎有三分之一的人每天在YouTube合計消費幾億個小時的時間觀看視頻，每日有一億支影片流量。接下來，全球資料量將急速攀升，到了 2025 年，全球的資料總量預計會達到 163 ZB 的程度，巨量資料也將為人類生活帶來前所未有的量化面向，進而使生活全然改觀。

* 1. **平行處理的效益增加**

摩爾定律（Moore's law）在1965年被高登·摩爾（Gordon Earle Moore）提出，指出[積體電路](https://zh.wikipedia.org/wiki/%E7%A9%8D%E9%AB%94%E9%9B%BB%E8%B7%AF)上可容納的[電晶體](https://zh.wikipedia.org/wiki/%E9%9B%BB%E6%99%B6%E9%AB%94)（晶体管）數目，約每隔兩年便會增加一倍。摩爾定律至今已經四十餘年，準確的描述了實際情況，至今仍被視為長期的趨勢。西元2000年後，由於CPU的時脈瓶頸，大部分CPU時脈在3-4GHz即無法再增加時，為了繼續提高效能，出現了多核心處理器（Multi-core processor）。多核心處理器較多個單核心處理器具有高主頻、、功耗低、通訊延遲低等優點。隨著摩爾定律的延續，多核心處理器的核心數將與日俱增。

2018年，主流的手機處理器的核心數目達到四核心，平行處理的效益也越來越高。

* 1. **使用CUDA進行GPU的平行計算**

GPU(graphics processing unit)一開始是為渲染圖形場景而設計，計算這些場景的座標等數據不需要以串列的方式一步一步執行，可以用並行處理的方式一次性渲染大量的數據，因此，從結構上來說，GPU不像CPU基於數個寄存器和高速指令集，GPU一般有數百個較小的處理單元。這些處理單元每一個都比CPU的核心慢很多，然而由於它的數目眾多，能夠同時進行大量運算，已經有越來越多的人嘗試用GPU來進行圖形渲染以外的工作，產生了GPGPU（general-purpose computation on graphics processing units)的概念。

1. **彩度採樣流程概述**

原始影像的檔案龐大，不利於儲存以及傳輸。在2015年Chien-Hsiung Lin, Kuo-Liang Chung, and Chun-Wei Yu的論文[1]中，基於數學最佳化的技術提出一種彩度採樣流程(Chroma Subsampling Strategy)，得到了對應的壓縮馬賽克影像的方法。

外界影像經過鏡頭擷取並透過感測元件後可以得到影像處理中的重點彩色濾波陣列(CFA, color filter array)檔案，而後再透過相對應CFA格式(Bayer等格式)的濾色陣列內插法(demosaicing)還原出完整的三原色(RGB, Red、Green、Blue)圖層，最後經由色彩編碼與壓縮的過程(YCbCr4:2:0等格式)完成整個影像壓縮處理的流程。

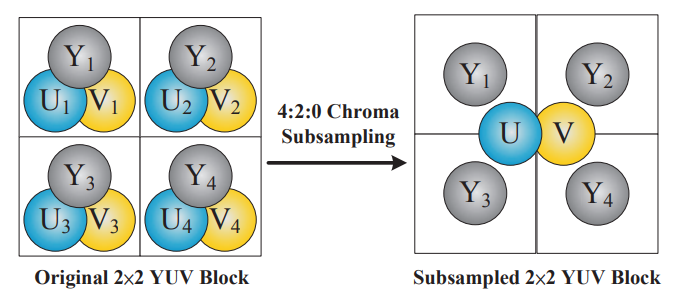
YCbCr 4:2:0影像處理流程為將還原後的三層RGB存檔，經由矩陣公式的運算轉換為亮度的Y層、藍色偏移量的Cb層與紅色偏移量的Cr層，此時得到的三層YCbCr存檔資料量與還原後的三層RGB存檔資料量相同，YCbCr 4:2:0影像處理架構是將Y層的資料完整保留但針對Cb與Cr層進行運算處理以減少其資料量，此步驟為彩度減少採樣流程(chroma downsampling)，經過此步驟後所得到的YCbCr 4:2:0影像處理存檔(.yuv檔)中的YCbCr三層資料相較於經RGB轉換後的YCbCr資料量比為1:0.25:0.25，其結果使整體的資料量縮小為二分之一。

1. **序列式壓縮影像流程**

接下來我們介紹一下我們實作的序列式(serial)去馬賽克運算優先壓縮影像(image)或是視頻(video)的流程。

對每一個欲處理的馬賽克影像，首先根據影像的CFA structure選擇去馬賽克技術來進行去馬賽克運算，得到去馬賽克的全彩影像(demosaicked full-color RGB image)。再由(1)式定義的RGB對YUV的轉換，得到YUV影像。

接下來，對的U與V進行4:2:0彩度減少採樣流程(chroma downsampling)。使對每一個的2 × 2 block，原先分別有4個U與V，採樣後只分別保留一個代表性的U與V，如示意圖1。



▲ 圖4：4:2:0彩度減少採樣流程[1]

最後，採樣後的YUV影像被送往影像編碼器(video encoder)，如H.264/AVC的編碼器或HEVC的編碼器，來壓縮影像(以4:2:0格式)。

當需要還原影像時，需要照壓縮時的相反步驟進行反向操作。首先，跟壓縮時對應的影像解碼器(video decoder)重新構造出採樣後的 YUV影像，此時每一個的2 × 2 block只有一個代表性的U與V，將代表性的U與V複製4份，使得每一個 × 1 block都有Y、U、V值。接下來，將轉換回RGB影像，轉換公式如公式(2)。最後根據影像的CFA structure選擇馬賽克技術來進行馬賽克運算即可。

當需要處理的是視頻(video)時，視頻其實由許多影像組成，我們流水線的分別處理每個影像即可。

1. **基於數學最佳化而設計的彩度減少採樣流程**

由於上文中提到的彩度減少採樣流程在每個2 × 2 block只保留一個代表性的U與V，因此會損失部分原本的影像資訊，所以總體而言，這並不是一個無損的壓縮，即還原的影像會伴隨部分失真。決定一個好的U、V值直接減少了失真的程度。因此，如果有一個策略可以選擇代表性的U、V值，我們就可以設計出更好的彩度減少採樣流程。

假設U和V為取樣後的樣本，因為品質失真的程度完全由U和V決定，因此我們把品質失真的程度量化，寫成函數的話，和就是影響函數值的變數，如下式。

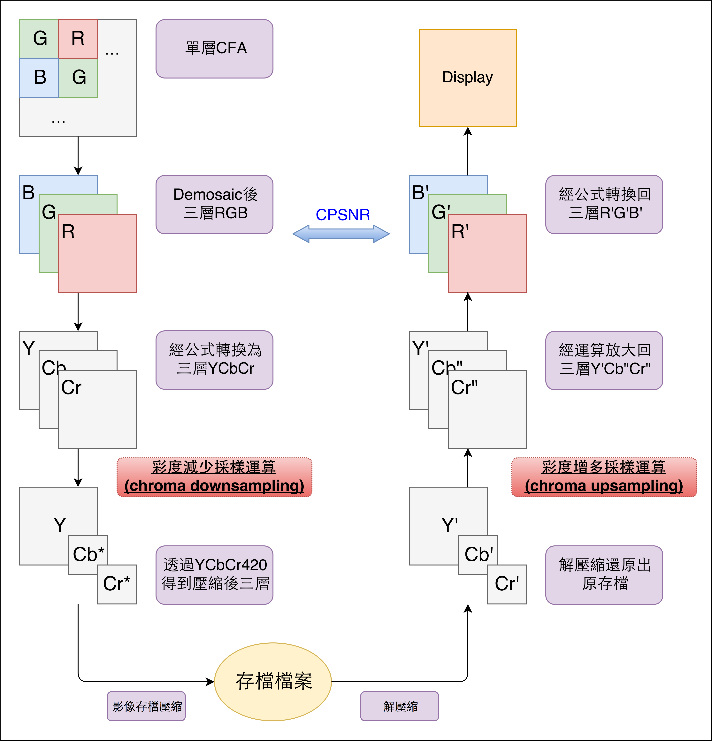
其中對每個，代表2 × 2 block的4個區塊各自的顏色失真程度。

經由文獻的閱讀理解，我們發現論文[1] 的整個流程中，一、RGB對YUV之轉換二、彩度減少採樣流程(chroma downsampling) 三、YUV對RGB之轉換。三處地方有進行平行加速運算的可能。需要補充的是，彩度增加流程(chroma upsampling)僅是複製U、V值，所以不需進行平行化處理。參考以下公式說明：

(1)與(2)為文獻中所使用的RGB對YUV以及YUV對RGB之轉換公式，藉由設定目標為使經YCbCr 4:2:0還原後之誤差能為最小而得到(3)。

將各項常數使用代數代換再展開矩陣運算可以得到(4)、(5)與(6)，最後將三式合併即得到文獻中的誤差公式：

在文獻的架構中提到，針對YCbCr 4:2:0的運算處理單位為2乘2大小，即表示式(7)在每單位中皆須進行獨立的運算，以單張圖片來看可能無太大的影響，因為輸入的資料還不夠大，但若將其放大至整部影片的處理，需要處理非常多資料，同時僅進行單一單位的運算對運行時間的影響是非常大的，故我們以此方向作為平行加速運算的出發點，目標能使對整部影片的處理時間有所降低。



▲ 圖5:YCbCr 4:2:0 流程圖

1. **Proposed Solution**

根據論文[1]提出的彩度減少採樣流程，我們可以用C語言寫出一個完整可執行的影像壓縮程式。接下來。我們用CUDA/C語言，將程式修改為可平行化執行的版本，使用CUDA來充分利用GPU的運算資源，達成GPGPU的概念。

影像和影片做平行化處理時，主要加速的部分在於矩陣運算。想要平行處理時，可能會遭遇到的問題如：工作的獨立性、工作的切割方式、以及資料的同步化問題等，針對上述問題，我們要對應的設計出平行化的方式。

平行化後的程式，需要滿足幾點條件：

1. 正確性：不論測資的大小，平行化後的程式計算結果與原來的程式相同。
2. 可延展性：較寬鬆的要求是當核心數目、執行序(thread)越多，也就是平行化的程度越高，執行時間越短。較嚴格的要求是運算資源增加倍，但資料量也增加近倍，執行時間並不增加，或是只些微增加。
3. 加速倍率：越高的加速倍率代表成果越好。加速比以如下公式定義：

{\displaystyle S\_{p}={\frac {T\_{1}}{T\_{p}}}}

其中：

指CPU數量

指順序執行[演算法](https://zh.wikipedia.org/wiki/%E7%AE%97%E6%B3%95)的執行時間

指當有p個[處理器](https://zh.wikipedia.org/wiki/CPU)時，[並列演算法](https://zh.wikipedia.org/wiki/%E5%B9%B6%E8%A1%8C%E7%AE%97%E6%B3%95)的執行時間

1. 效率：由加速比衍生出的效率（英語：efficiency）則是量度效能的指標:

效率的值一般介於0～1之間，用於表示在解決問題時，相較於在通訊與同步上的花費，參與計算的處理器得到了什麼程度的充分利用。

經由此次研究，我們可以對平行化程式在不同運算資源時的運算時間進行統計，並進行分析，藉此得知是否有改進的地方，或是什麼類型的運算資源較有優勢。

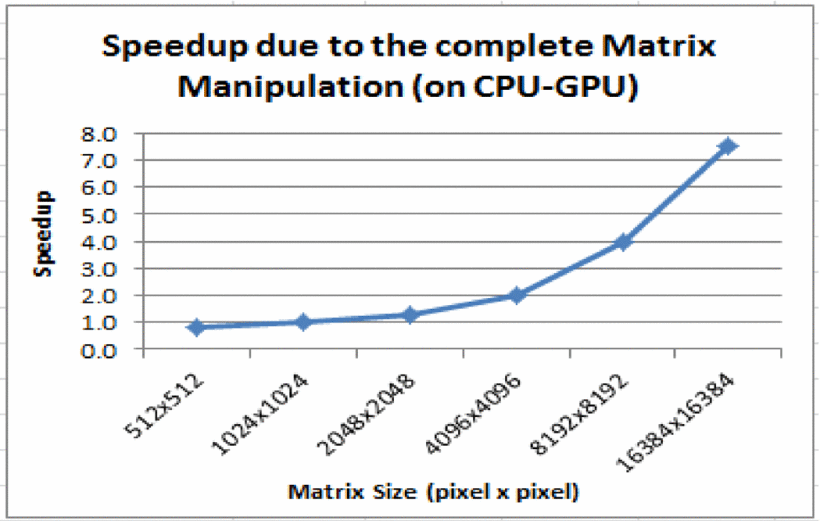
我們預計選擇CUDA進行平行運算加速，使用CUDA進行是考量到若使用CPU僅可增加運算的執行續達到加速的目標，而CPU目前較高階普遍僅有8~16個執行續，因此我們認為使用具有更多運算單元(預計使用GTX 1080Ti具有3584個CUDA Core)的顯示卡進行加速運算更能達到我們預期的效益。

參考文獻內容，文獻使用Visual Studio C++ 進行整個流程的建立，我們初步計畫使用與文獻相同之環境進行加速優化，預計使用的環境為Visual Studio Community 2017，未來也可能會視進行的情況進行修正。

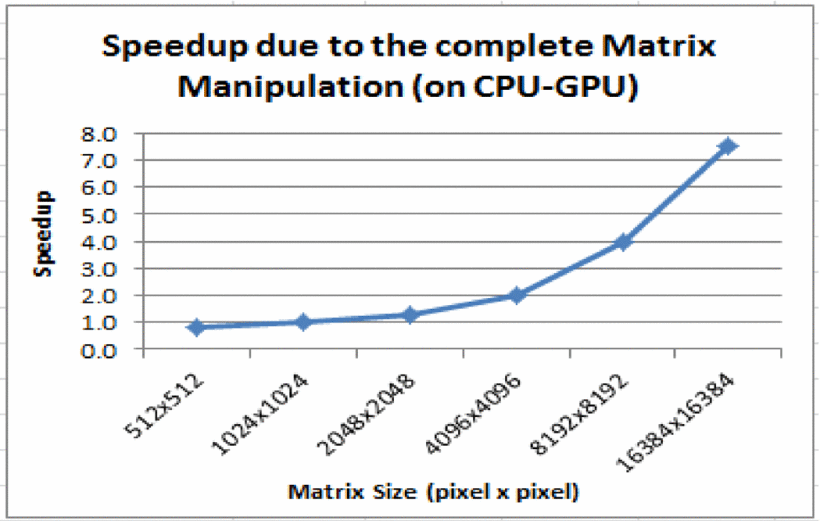
1. **Experimental Methodology**
2. **Experimental Results**
3. **Related work**

Helly M. Patel et al於2015年發表論文[5]，他介紹了大規模的影像處理可以藉由Hadoop、 CUDA或MPI進行平行運算。但其並未進行實際實驗，只比較了各種平行運算的優劣利弊。

[Abu Asaduzzaman](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22First%20Name%22:%22Abu%22&searchWithin=%22Last%20Name%22:%22Asaduzzaman%22&newsearch=true&sortType=newest)et al於2015年發表論文[6]，主題是用CUDA進行影像處理的平行運算，他們分別撰寫了序列式(sequential)的C語言程式與平行的CUDA/C語言程式，並計算加速比(speedup)。若只考慮實際有平行化的部分，即矩陣運算的部分，加速比甚至高達數百倍，如圖，不過若考慮整個影像處理程式的執行時間，加速比最高接近八倍，不過要在圖片畫值達到極高的16K才行，在目前已經算很高的4K畫值加速倍數僅2倍，如圖。值得一提的是，他們指出CUDA的host至device的複製運算(copy operation)造成了可觀的間接費用(overhead)對大的影像來說CUDA的malloc()操作消耗的時間可以忽略，但是將資料從CPU複製到GPU的時間影響了加速倍率。

****

▲ 圖6：矩陣運算加速倍數[6]

****

* 圖7：影像處理加速倍數[6]

Tse, Jia Jun 於2012發表論文” Image Processing with CUDA”[7]，詳細介紹了GPU以及CUDA的原理，在影像處理方面他考慮高斯模糊(Gaussian Blur)以及基於索伯算子的邊緣檢測(Sobel Edge Detection)這兩個運算，以CUDA/C語言寫出程式。

|  |  |  |  |
| --- | --- | --- | --- |
|  | GPU Time(ms) | CPU Time(ms) | Percent Increase |
| 512 x 512 Lena | 0.67 | 16 | 2,288 |
| 1024 x 768 wallpaper2 | 0.84 | 62 | 7,280 |
| 3200 x 2400 cartoon | 2.92 | 688 | 23,461 |

* 表格1：高斯模糊的實驗結果[7]

|  |  |  |  |
| --- | --- | --- | --- |
|  | GPU Time(ms) | CPU Time(ms) | Percent Increase |
| 512 x 512 Lena | 0.67 | 32 | 4,676 |
| 1024 x 768 wallpaper2 | 0.82 | 94 | 11,363 |
| 3200 x 2400 cartoon | 2.87 | 937 | 32,548 |

* 表格2：索伯算子的邊緣檢測的實驗結果[7]

可以看出，與我們的結果相同，需要運算的資料量越大，CUDA運算的潛力也越大。

1. **Conclusions**

我們完成了論文[1]內容的程式，與程式可平行化執行的版本。利用平行化的技術，充分利用了現在普遍的多核心電腦系統，或是多台電腦同時運算，使平行化的程式較原本的程式快，並具有可延展性(scalable)的特性，即平行化的程式運算資源越多，執行的越快。並且，當平行化的程度越高，加速倍率也越高。

這篇論文對CUDA的加速效益提出了實際數據，不過我們仍有改善的空間。實驗的影像資料仍然可以加大，對記憶體的分配仍然可以改善。甚至，雖然CUDA已經可以做出商業化的產品，但他還算是一個年輕的語言。Nvidia最新的Fermi­—次世代 CUDA架構，具備多達 512 個 CUDA 核心，並且可以平行運行多個核心程序(kernel)。隨著科技進步，產品也會越來越好。

1. **References**
2. C.-H. Lin, K.-L. Chung, and C.-W. Yu, "Novel chroma subsampling strategy based on mathematical optimization for compressing mosaic videos with arbitrary RGB color filter arrays in H.264/AVC and HEVC," IEEE Transactions on Circuits and Systems for Video Technology, vol. 26, pp. 1722-1733, 2016.
3. W.-J. Yang, K.-L. Chung, W.-N. Yang, and L.-C. Lin, "Universal chroma subsampling strategy for compressing mosaic video sequences with arbitrary RGB color filter arrays in H.264/AVC," IEEE Transactions on Circuits and Systems for Video Technology, vol. 23, pp. 591-606, 2013.
4. T.-L. Lin, P.-S. Liaw, S.-L. Chen, C.-H. Hsia, and S.-Y. Lin, "Chroma 422 subsampling for bayer pattern in H.264 video coding," Electronics Letters, vol. 52, pp. 608-609, 2016.
5. H. Chen, M. Sun, and E. Steinbach, "Compression of bayer-pattern video sequences using adjusted chroma subsampling," IEEE Transactions on Circuits and Systems for Video Technology, vol. 19, pp. 1891-1896, 2009.
6. Helly M. Patel, Krunal Panchal, Prashant Chauhan , M. B. Potdar, “Large Scale Image Processing Using Distributed and Parallel Architecture,” International Journal of Computer Science and Information Technologies, Vol. 6 (6) , 2015.
7. [Abu Asaduzzaman](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22First%20Name%22:%22Abu%22&searchWithin=%22Last%20Name%22:%22Asaduzzaman%22&newsearch=true&sortType=newest) , [Angel Martinez](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22First%20Name%22:%22Angel%22&searchWithin=%22Last%20Name%22:%22Martinez%22&newsearch=true&sortType=newest), [Aras Sepehri](https://ieeexplore.ieee.org/search/searchresult.jsp?searchWithin=%22First%20Name%22:%22Aras%22&searchWithin=%22Last%20Name%22:%22Sepehri%22&newsearch=true&sortType=newest) “A time-efficient image processing algorithm for multicore/manycore parallel computing,” Published 2015 in SoutheastCon.
8. Tse, Jia Jun, "Image Processing with CUDA" (2012). UNLV Theses, Dissertations, Professional Papers, and Capstones. 1699.
9. 周秉誼, “GPU高效能運算環境—CUDA與GPU Cluster介紹”, <http://www.cc.ntu.edu.tw/chinese/epaper/0012/20100320_1205.htm>, 2010
10. H. Chen, M. Sun, E. Steinbach,“ Compression of Bayer-pattern video sequences using adjusted chroma subsampling,” IEEE Trans. Circuits and Systems for Video Technology, vol. 19, no. 12, pp. 1891–1896, 2009.
11. K. Hirakawa and P. J. Wolfe, “Spatio-spectral color filter array design for optimal image recovery,” IEEE Trans. Image Processing, vol. 17, no. 10, pp. 1876–1890, 2008.