影像處理作業

Spatial Domain JPG

An "interesting" new image compression technic based on median filter, color space transform, and BRUTE FORCE.

系級：資工四 A

學號：107502520

姓名：黃允誠

2020 年 1 月 15 日

※因為怕萬一 word 到教授那邊格式跑掉，所以我同時附上了 pdf 以備不時之需。

介紹

動機

在剛開學沒多久，投影片第二章教授有說過：「空間解析度跟灰階解析度在同樣資料量限制下要取捨，簡單影像優先選色彩解析度，複雜影像優先選空間解析度。」

我當時就一直覺得這句話很有趣：雖然說教授放了幾張範例，從簡單到複雜的影像，但實際上一張影像很可能同時有非常單調的部分，另一邊又有非常細碎需要高空間解析度的部分。我當時就在想，如果一張影像，某些地方空間解析度高，某些地方空間解析度低，可不可以呢？

所以之後沒有想到其他更有趣的題目，我就決定做這個了。至於為甚麼最後變成在做壓縮……且聽我娓娓道來。

構思過程

在剛開始的時候我想法很單純，就是某些地方的 pixel 大一點，某些地方小一點，然後另外用某種方法紀錄。但是很快就卡關了，之所以圖片的空間解析度通常整張一樣的原因，就是不一樣的話會像在整理家裡的房間一樣，根本不知道要怎麼把全部的東西拼好。

就在這時我忽然又想到另一個問題：我又不會畫畫，就算真的讓我設計出一個這種很好的檔案格式，我要怎麼表現？這時候很容易就會想到拿網路上的圖片來用，結果思路不知不覺就從「設計同一張圖片不同空間解析度的表達方式」變成了「從一張圖片挑一些區塊合併成同一個 pixel」，最後就跑去做壓縮了……

於是我開始思考怎麼從圖上找出「較簡單的部分」。上完教授一整學期的課之後再回去看那句話，很快就會想到那一堆轉成頻率域的公式，頻率高不就是複雜嗎？但是這樣就落入陷阱了，我的目標是在圖上找出簡單的部分，合併空間複雜度的同時不去動到複雜的部份，教授說過在頻率域的動作是「牽一髮動全身」，一旦轉到頻率域反而束手無策。這就是這個方法相比 JPEG 唯一的優勢了，最後再提。就算我想局部做轉換，到底要在哪個地方、多大的區域做轉換？就算用一個遮罩直接暴力一步一步做區域轉換，那麼找到頻率很低的地方之後，又要開始煩惱到底要用多大的遮罩？理論上我會希望盡量找到更大的區域一次收縮，那找到一個頻率低的地方之後，難道從 3x3 開始慢慢放大遮罩一直試，不斷的做不同範圍的傅立葉轉換嗎？光想就很恐怖，既然這樣，乾脆就在空間域暴力搜索！

原理

核心概念

首先從左上角開始往右往下走，跟一般的區域/點處理類似，但其實從最左上第一個點就開始找了：對每個點往右往下，從近的點開始判斷，找出所有跟它很像的點最後框成一個矩形，全部當成同一種顏色。這個矩形視乎圖片本身的 pattern 可能很小但也可以很大，甚至可以大到直接佔據一小片螢幕(例如在夕陽照片中的太陽上面)，只要在人眼沒那麼容易發現的情況下就好。

那麼第一個重點來了，甚麼叫做"很像的點"？一開始當然想說比大小就好了，但後來忽然想到教授說過直接用 RGB 處理往往效果不好，我決定轉到 YCbCr，主要是因為有教授教的精簡版本轉換公式減少計算量。

第二個重點就是，從上面應該不難看出我想做的事情會受到「胡椒鹽雜訊」很嚴重的影響，本來大家都差不多，正想框起來的時候中間有一粒胡椒鹽壞了一鍋粥，剛好我記得中值濾波器很能處理這個，所以就先用一次中值濾波器，明顯發現能框的盒子平均大小提升。

實作細節—關於判斷框框內的點是否很像

第一就是 Y 代表亮度，CbCr 代表色度，所以人眼對 Y 比較敏感，換句話說在判斷是否夠接近的時候，Y 的門檻值設比較嚴格。

第二就是我透過好多次實驗發現，人眼除了對亮度較敏感之外，在亮度較暗的時候，不管對亮度還是色度都比較遲鈍，亮度較亮的時候比較敏感。例如一片黑漆漆的夜晚海面，顏色看

起來都是一片黑，但白天太陽旁邊的雲朵，我程式只是稍微多框了一點框框，馬上就很明顯看到了。這時候轉過來的好處就體現出來，正好直接拿 Y 來判斷，越暗的時候亮度色度差距的門檻值都可以調高。

第三就是我的作法並不是單純一個一個點判斷，而是再用一個迴圈慢慢擴大框框，每次擴大的時候，要求 YCbCr 三個數值分別都要足夠接近，而且是用所有框框內的點的最大值跟最小值比！三個數值分別都各自挑出所有點裡面的最大最小，還要求夠接近，所以框起來基本上裡面的顏色全部都非常接近。

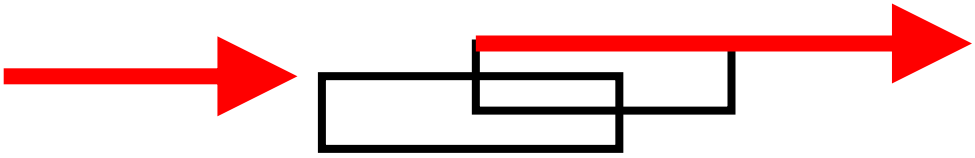
最後稍微提一下一個比較遺憾的，我本來把反轉換也寫好了，而且還發現一個很酷的方法，就是 YCbCr 都用 1 byte 存，但是 Y 用無符號，CbCr 用有符號，這樣它們可以擠在 3 byte 而且都不會溢位！教授說的那句話我其實只做一半，就是「簡單的地方空間解析度降低」，而另一半我本來想雙管齊下，就是框框很小的話(代表是影像比較複雜的地方)，就減少灰階解析度，例如把低的 4 bit 砍掉然後兩兩擠在一起，結果我不管怎麼弄都會變成很明顯的彩色斑點，不管壓 CbCr 還是 Y 都是。明明記得說色度比較可以壓縮啊！我上網查了一下資料，說是「抽樣」，結果仔細看原來也還是在空間上面作手腳，這個我現在要做的作法已經做得夠徹底了……可能如果要減少灰階解析度，必須要一開始就決定好，不能對已經拍好的影像這樣硬來吧……真可惜。

實作細節—中值櫻吹雪

本來我想說 3x3 已經是最小的，結果邊都快不見了……這時忽然突發奇想，只看上下左右一共五個點取中位數，好很多。後來查資料才知道原來大家本來就這樣做的，說是十字中值率波，但我還是覺得看成一堆花比較浪漫！

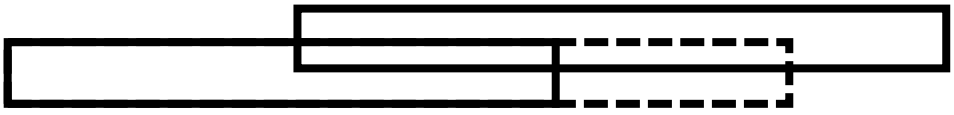
實作細節—這樣會不對稱嗎？

我現在用的拉框框方式，是先往右下角拉盡量大的正方形，然後開始往右跟往下找，看往哪邊拉出的長方形長度最長就選哪邊。教授可能會覺得從前面看到這邊為止，每一個步驟都非常地不對稱，但其實這個問題沒有這麼嚴重。首先已經被框過的點會有紀錄，後面直接跳過沒錯，但這種一條一條掃描的方式，就會產生一種後面框框撞前面框框的情況，像這樣：



我本來的作法是，既然前面的點之前已經框過了，那這邊撞到就停下來吧，結果發現變成都傾向拉很長的縱向框框。接著我很直覺的把最外層迴圈調換，就是變成先往下再往右的遍歷，果然變成都傾向拉橫向的細長框框，就像教授可能擔心的一樣。詳細的原因不太好解釋清楚，可以簡單理解為鋪地磚的時候，沒有對齊就會越來越錯開，畢竟我本來也就沒要求框框之間要對齊，而因為遍歷的方向性，導致其中一個方向特別容易越來越錯開。

而如果直接就不管它繼續拉，那變成後面的框框蓋掉前面的框框，一樣不太公平，最後我選擇折衷方案：



像這樣，撞到的時候紀錄撞到的位置，並繼續看本來要框到哪裡，最後框到中間一半的地方。要知道雖然這兩個框框都想框中間那個區域，但那只代表它們跟那個區域都還算接近，兩個框框顏色還是有點差距，這樣的做法有點取平均的味道！

最後，一個框框內的顏色當然也不是挑左上那個點，而是最後框完之後挑矩形中心的點。

結果

.sdjpg 檔案

講了那麼多，都還沒講到我壓縮到底壓縮成甚麼。在 data 資料夾裡面教授會看到許多電腦認不得的檔案，這是我用 c 的檔案讀寫函式，基本上隨便自己打任何副檔名都可以。那這個檔案裡面到底是甚麼格式？其實非常簡單暴力，首先最前面是兩個無符號四位元整數，代表圖的寬度跟高度，剩下就全部都是框框了。

每個框框也要記錄它的長寬，還有顏色，但框框的長寬最多基本上都不到 255，我也有設檢查，為了極少數極端情況多留空間不值得。所以就是每個框框 5 個位元，這樣一直下去，框框可大可小，但都只要 5 個位元就能記錄，以此完成壓縮。

那框框的位置呢？檔案的結尾呢？這就是最酷的地方，只要一樣用這套程式，它算出來的框框，一定也能用一樣的方式剛好填滿圖片。所以解壓縮的時候就是一樣再來一次，照著順序碰到還沒處理的點就把框框放上去，放了之後已經被框框佔據的點就跳過，找下一個放框框的點，直到碰到圖片大小最右下角檔案也剛好讀完，就這麼簡單！

壓縮的成果

那些.sdjpg 檔雖然沒辦法直接開，但還是可以查看檔案大小。跟原本的 bmp 點陣圖比基本上都是壓到剩 5～10 分之一，以初學者的作業來說應該算不錯了……吧……

如果教授想看壓縮過的圖片到底長怎樣，可以直接點檔名加了_decomp 的解壓縮後檔案看，放到很大的時候在平滑處會看得到有點一塊一塊的，但平常肉眼是難以看出來的。

程式

有兩個程式，執行之後都是輸入檔案名稱壓縮或解壓縮，沒有打副檔名程式會自己補上.bmp 或.sdjpg，只要目標檔案跟執行檔在同一個資料夾就好。教授要求程式碼跟圖片分開資料夾放，所以如果想要試試看記得要另外放一起。然後那個「superBIG!!!」不要輕易嘗試，它是特別大，我在網路上特別找的，找到最大的，要跑比較久……

程式碼的部分，雖然上面看起來講概念很輕鬆，但實際寫的時候好多細節，壓縮器中間有一段特別醜特別雜的，就是我在處理框方框的事情……我真的已經盡力寫得比較好看了 Orz

沒有甚麼輔助說明的圖片

基本上放在資料夾的就是全部了，每個圖檔都有對應的壓縮解壓縮之後的圖檔，我看起來是都還蠻順眼的，沒有甚麼明顯的瑕疵。

討論/結論

(現在)打不贏 jpeg……

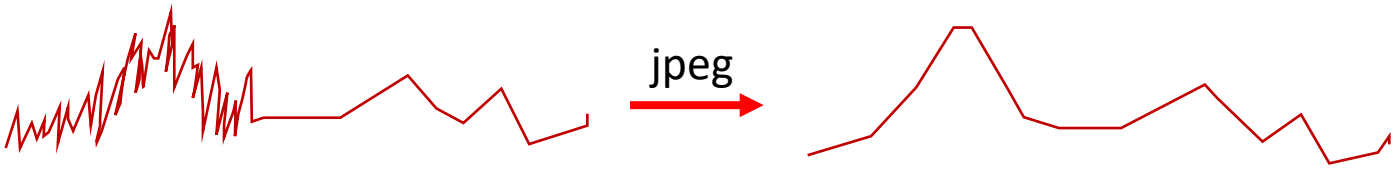
這個結果差 jpeg 還是有一截……其實本來我就像最前面說的，是受教授投影片上那句話啟發，單純有這樣一個想法實作出來而已，並不是受 jpeg 啟發，而這個附檔名跟報告題目也是後來想的。

那為甚麼忽然想到 jpeg 呢？因為在實做的過程中，越做越覺得好像看起來跟 jpeg 壓縮的圖片有點像……所以我就去了解了一下 jpeg 大概的壓縮原理跟流程。然後我才忽然發現真的非常像！雖然說是像，但其實還是有一些差別：

首先第一點，jpeg 主要是轉到頻率域之後做壓縮，就像最前面提到的，這樣的做法雖然很強，但相對的也較沒有彈性。我的這個方向，直接在空間域上做，雖然很暴力耗時也較久，但確實可以針對當下圖片的特性去做更大程度的因應。

接著第二點，jpeg 的壓縮流程中，最大幅度的壓縮都集中在最後的 Z 字形跟霍夫曼編碼還有省略一堆零那邊，這些技術我也還沒有用上(應該說現在的我其實也還想不到怎麼用……)。

最重要的第三點，如果仔細想會發現 jpeg 跟我做的事情看起來很像，實際上幾乎是相反的！雖然兩者都是會讓圖變得平滑，過度壓縮也都一樣會出現大塊肉眼可見的區域，但是 jpeg 轉到頻率域之後，它的理論是人眼對高頻信號比較難辨認，所以它是挑高頻的地方壓縮，但我反而是挑低頻的地方下手！用文字說明可能還是有點模糊，我試著用圖解描述這個概念：

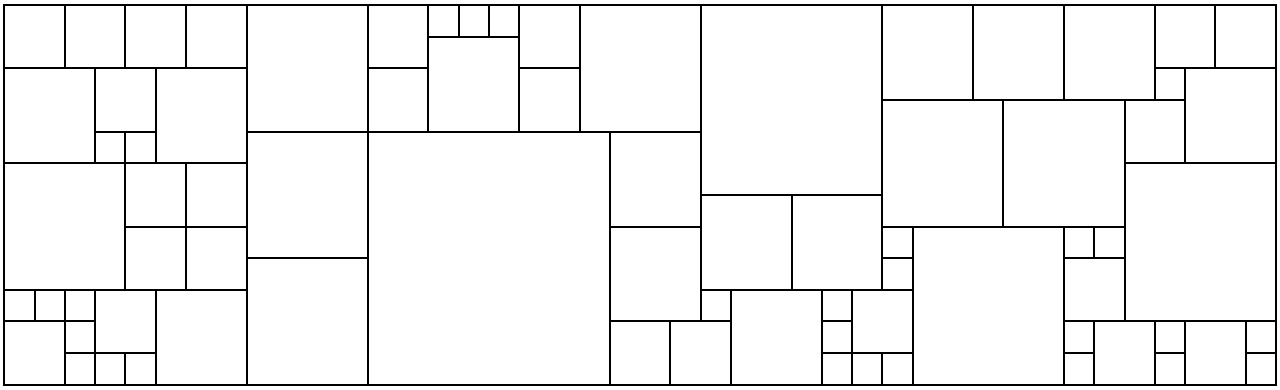


這就代表這個方式不一定要跟 jpeg 當競爭者，未來有可能可以找到某種方式，讓兩個技術一起並行相輔相成！

最後再提兩個我有想到可能的改進方向，主要是沒時間做了：

第一，我用的擴充框框的方式還不夠聰明，我有想到有些時候影像上的光影變化是有方向性的，例如海邊看夕陽的照片，光影垂直方向變化會比較大，水平方向會比較少，甚至水面上的倒影都是橫條紋，這時候用橫向細長框框可能反而能框出更大面積。我們可以使用差分微分那一類的方式，先推估出圖片在這個位置光影變化的方向，並作為畫框框的依據。

第二，其實我本來在猶豫兩種做法，另一種我沒做的是：在圖像上劃分大小不一的、邊長是 8 的倍數的正方形，然後正方形內做空間解析度的收縮，例如說邊長 16 的正方形，就每四個像素併成一個。問題在需要一個方式判斷到底這邊多大的正方形可以執行這個操作，又不會造成肉眼可見的影響。這個方式的重點在於，最後產生的都是一堆大小不一、但全部都是 8x8 矩陣的影像，剛好可以直接給 jpeg 用它的流程壓縮，這樣兩階段的方式直接將我這個空間解析度的想法與 jpeg 成功串接。以下圖例，最小格子代表沒能收縮的 8x8 像素區塊：



總之即使不意外輸給 jpeg 我不怎麼失望，雖然目前我的程度沒法體現這技術的威力，但我相信它是非常有潛力的新方向。