國立台北大學資訊工程學系專題報告

相似系列圖片自動整理系統

專題編號 PRJ-NTPUCSIE-110-004

執行期間: 2021年9月 至 2022年5月

一、摘要

因智慧型手機具備的相機功能及其與社交APP的緊密連結，相片與圖片的累積就成爆炸性地增長，當檔案數量多到難以親自整理的程度，便會想依賴程式的力量，本專題便以「檢測重複相片」為出發點，做出一個在電腦端執行的相似系列圖片整理系統。

二、簡介

現在大部分使用者對於圖片的使用習慣，常常是將許多圖片儲存到同一資料夾中卻無法將其好好分類，例如會將偶像的重複照片存到裝置內的追星族，或是在日常對話中會加入梗圖的梗圖使用者，都會有圖片過於雜亂且未經過整理而找不到圖的窘境。

在蒐集市面上檢測重複相片的電腦軟體後，我們發現其檢測功能都有些不足。撇除基本的相同照片檢測不說，一般的此類軟體到相片被進行一些基本修改，例如裁切、調色等等，就無法檢測出來，因此我們決定針對這類功能進行研究，並嘗試加入能夠檢測裁切、調色過的相片的功能，力求讓使用者能利用程式將相似圖片(如圖一～四)快速分類，在這方面的軟體使用體驗上更加優化。

圖一、部分相同或經裁切的圖片

圖二、人物背景、姿勢極相似的圖片

圖三、調整過色調、濾鏡的圖片

圖四、文字代換的、有模板的梗圖

三、相關文獻探討

3.1 aHash算法[1]

均值哈希(aHash)算法是一種圖形相似度識別的演算法，將圖片壓縮到一定尺寸並將RGB轉換為灰度值，這樣可以提取圖片整體基本結構以及明暗等訊息並去除圖片上較為複雜的細節雜訊，全部縮為一定尺寸也可保證哈希值(hash)為相同長度便於計算對比。

算法步驟:

1. 將圖片像素壓縮成8*8。
2. 將處理後的圖片灰度化並算出均值。
3. 將每個像素的灰度與前述之均值進行比較，大於等於均值計為1，小於均值計為0，得出一組64(8*8)位之01字串(布林值)。
4. 將圖片的64碼布林值一一配對並計算彼此的漢明距離，若漢明距離小於設定值將兩圖片視為相似。

3.2 SIFT演算法[2]

尺度不變特徵轉換(Scale-Invariant

Feature Transform, SIFT)是一種電腦視覺化演算法，能在空間尺度中找尋極值點，並取其位置、尺度(向量)、旋轉不變量等等，主要是用來偵測與描述圖像中的局部性特徵。

SIFT演算法會利用不同尺度空間上找出其關鍵點並計算出關鍵點向量，不易受到縮放、旋轉、光照、噪音等因素變化而受到影響，故這些關鍵點在圖像上具顯著性。

SIFT算法步驟**:**

1. 高斯模糊：先將原始圖像與N個不同倍率的高斯模糊函數進行卷積產生尺度空間，將這些尺度空間集合成一組且有N層(N通常為6)。

*二維空間高斯模糊函數

$$G(x_i, y_i, \sigma) = \frac{1}{2\pi\sigma^2}\exp\left(-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma^2}\right)$$

定義原始圖像***I(x,y)***與一個可變尺度二維高斯函數***G(x,y,σ)***卷積運算。圖像尺度空間表達式：

$$L(x,y,\sigma)=G(x,y,\sigma)*I(x,y)$$

2. 圖像金字塔：將原始圖形經不斷地降維(縮小)利用高斯模糊函數進行卷積型成多組的照片集合，進而構造成圖像金字塔，圖像金字塔的組數是由圖像大小所決定。
3. **DOG(**difference of Gaussian**)**：得到的圖像金字塔轉換成高斯差分金字塔，將每一組的相鄰兩層進行相減得之。
4. **極值檢測：為尋找極值點會利用高斯差分金字塔上的每一層的**內部組數近行比對，內部層數上的每個點會跟自身周圍的8個點比以及上下兩層的9個點比，若與鄰近的26個點(8+9+9)相比為最大或最小者會被稱之為”暫定”關鍵點。
5. 關鍵點定位：由上步驟得到的關鍵點為暫定的，這主要是因為那從離散空間上取得的並非最終所需，需進一步找尋最終的關鍵點;將這些暫定關鍵點運用在數學座標上利用導數後進行泰勒展開取的一個連續函數，再從連續函數後透過微分得到極值點。
6. 消除邊界響應：因一開始使用高斯模糊函數，故可能增強圖形邊界數值所以增加此步驟降低便界的影響力，使用Hessian矩陣進行弱化。
7. 定位主方向：經Hessian消除邊界響應後的得到更精準的關鍵點，接下來每個關鍵點會使用到4*4共16個種子點來描述(種子點是由4*4矩形形成)，種子點內部會計算出八個方向(45的倍數)的量值共會得到128個方向(16*8)，故一個關鍵點會得到128維度向量，再把各個角度的值相加得出最大的那個角度為關鍵點主方向。

四、研究方法

4.1 兩圖片檢測

起初，我們認為比對圖片最基礎的功能應是比對整張圖片的功能，於是我們將aHash用python實現出來，發現aHash對於檢測完全同圖的效果極佳，速度非常快，也可處理色調、色溫、濾鏡等等的比對，以90度為基準的旋轉以及鏡像也皆可處理。

但aHash的致命缺點就是完全辨別不了經裁剪過或部分相同的圖片(如圖五中左、右兩圖皆為圖六之子圖且有重要資訊重合)，在本專題中我們將之稱為相似系列圖。

為了檢測相似系列圖，首先我們將研究重點放在了如何將兩張部分相同的圖片中的「相同區塊」檢測出來並且進一步用aHash全圖比對出來。

圖五、左圖 size1-1 2806X2078
右圖 size1-2 1722X1340

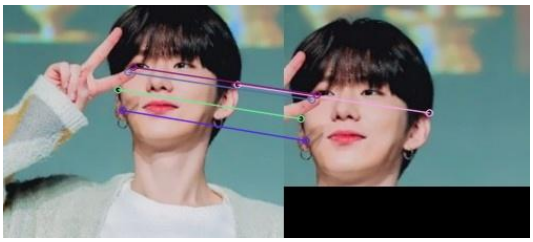
圖六、完整圖片

方法步驟：

1. 使用SIFT演算法將兩圖片中的關鍵點取出，如圖七。

圖七、取關鍵點之圖片

2. 使用BFmatcher比對兩張圖片中最為相似的五個關鍵點，如圖八，以此分別在兩張圖片中圍出C_3^5個三角形。取五個關鍵點的用意是在能容許一些在兩張圖片加上相同浮水印而造成物的誤判，因時間上的考量才決定採用五個關鍵點至多做十次比對即可，而不是取越多越好；若採取五個相似關鍵點能允許內部最多兩個關鍵點是被浮水印影響，至少還能剩三個關鍵點判定圖片是否相似。



圖八、相似關鍵點連線

3. 檢查是否存在兩個相似三角形。為此我們確認兩個三角形長寬比與角度比，若長寬比不一致則直接判斷不是相似圖片，若角度不相近也直接判斷不是相似圖形。
4. 若能找到兩個相似三角形，我們截取兩圖片中相似三角形延伸出去的長方形區塊，算出倍率，進行縮放為相同尺寸，例如:(參照圖五)

   經計算size1-2是size1-1 的 0.5001571106635023 倍
   ⇨ 將size1-1縮小

圖九、經截取並延伸長方形示意圖

縮放完後，並計算哪張圖面積較大　並變作為後續分群代表圖，簡稱為「母圖」。

5. 將截取並縮放後的兩個長方形圖片區塊(如圖九)進行aHash比對，若比對出來的漢明距離小於給定距離，則我們將兩原始圖片視為"相似系列圖"，分在同群，並將「母圖」置於群首，例如：size1-2較size1-1圖片完整度較高，故將size1-2視為這個群的母圖，置於群首。

此外，基於這基本架構，我們還增加了Size-Reduced步驟改進檢測效率，將原始圖片長寬都超過1000的縮小2倍、寬都超過2000的縮小4倍、長寬都超過4000的縮小8倍，而經縮小過後算取關鍵點、關鍵點向量、兩圖片的BFmatcher時間大大降低且並不影響準確率。

4.2 多圖片檢測

在圖庫比對中我們則將最原始的C_2^n的對比次數用群「母圖」的方法將其減少，亦即，在比對的過程中我們會記錄各個被歸類在同一群的圖片中何者為最大圖(場景區域最大)，後續的圖只需要與每個群的最大圖進行比對，即可知道是否屬於此群，省去了與多餘圖片比對的時間。

圖十、左上、左下、右，3張系列圖

然而，這樣的方法其正確性將會嚴重受制於圖片的檢測順序，若真正的母圖在許多子圖分群後才出現，則會出現應該分成一群的子圖被分成不同群的問題，例如，假設圖庫中依序有圖十中左上、左下、右三張圖，則按上述方法，分群結果將會是圖十之左上、右兩張圖為第一群，圖十之右圖為該群母圖，圖十之左下圖為第二群。

為了解決這個問題，圖庫比對的過程中，若一群母圖發生改變，則需繼續進行新

母圖與往後其他所有群母圖的檢測。

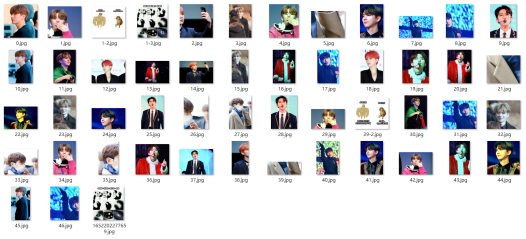
圖十一、群母圖發生更換，產生群合併

完整圖庫檢測步驟：

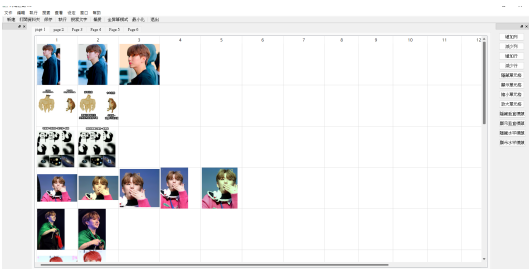
1. 一新進圖片跟一群母圖比較，若非相似系列圖，則繼續與下一群母圖比較，直至發現相似系列圖，或無群母圖可比較，則新圖自成一群，新圖為該新群之母圖。
2. 發現新圖之相似系列圖群，若新圖較母圖為小，新圖加入群後方。
3. 發現新圖之相似系列圖群，若新圖較母圖為大，新圖為新母圖，新母圖在往後與剩餘的每個群母圖比對，若後續群母圖與新母圖為相似系列圖，則將兩群合併，新母圖仍為該群母圖，如圖十一所示。

五、專題實作

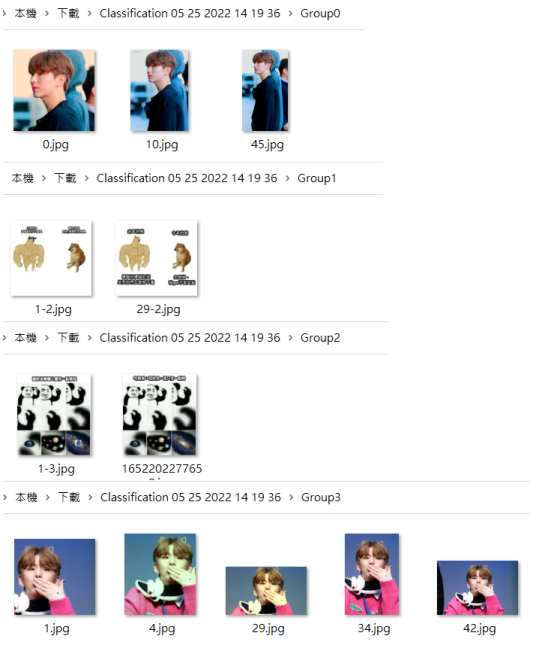
使用python的pyqt5[3]套件最為前端開發，搭配使用OpenCV[4]作為圖像處理的工具。建置的軟體主要功能為：可對使用者選擇之資料夾（如圖十二）進行掃描，將混亂的照片分群並顯示（如圖十三），使用者亦可將分好群之照片匯出，一群一個資料（如圖十四）。



圖十二、資料夾包含混亂照片



圖十三、軟體分群結果



圖十四、分群照片自動存到不同資料夾

六、實驗數據

6.1 實驗設定

我們準備了兩大類資料集做測試，第一類資料集包含100組圖片，每一組為一張

原圖及被修改過的圖，共200張圖片；第二類資料集包含2張一群*8、3張一群*8、4張一群*8、5張一群*8、6張一群*8、7張一群*1，共有41群167張圖片，測試時皆已打亂。

以missed rate（MR，屬於同組/群，未被檢測出）與incorrect matched rate（IMR，不屬同組/群，被錯誤歸類）評估相似系列圖片檢測能力：若兩張圖片應為一組/群，則將其關係以布林值1表示，反之，以布林值0表示。以G表示實際關係，T表示檢測關係，則定義為：

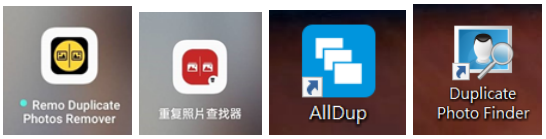
$$MR = \frac{\#G1 \cap T0}{\#G1}$$

$$IMR = \frac{\#G0 \cap T1}{\#G0}$$

6.2 實驗比較

6.2.1 軟體比較

首先，我們以第一類資料集比較我們的軟體與既有軟體對系列圖片的檢測能力，比較的軟體如圖十五所示，包含手機APP（A、B）跟電腦程式（C、D）。



圖十五、由左至右為軟體A、B、C、D

表一、軟體能力比較結果（組）

軟體	A	B	C	D	Ours
原圖 vs 原圖	63	77	90	100	100
原圖 vs 子圖 (裁切後的圖片)	0	0	0	0	99
原圖 vs 原圖色調	21	24	88	5	99
原圖 vs 子圖	0	0	0	0	98

表一結果顯示，我們的軟體在原圖檢測中優於既有軟體，在檢測系列子圖及色調圖方面，能力亦相當優秀。

6.2.2 方法檢測能力評測

我們以第二類資料集評測我們的系列圖檢測演算法，比較的方法包含：

Ours - 先對初始圖片做Size-Reduced，再使用4.1方法檢測配對圖片，並以4.2方法檢測圖庫。
A - 用aHash做暴力比對。
B - 基於4.1方法，但以match-template 取代步驟4，不做步驟5，用暴力法做圖庫檢測。
C - 先對初始圖片做Size-Reduced，再承接方法B。
D - 採4.1基本方法，暴力法做圖庫檢測。
E - 基於4.1方法，但以match-template 取代步驟4，用暴力法做圖庫檢測。
F - 類似Ours，但以match-template 取代4.1步驟4，且用暴力法做圖庫檢測。

表二、大圖與子圖比對結果[1][2]

	Off-Time	On-Time	MR	IMR
A	3.475 sec	0.075 sec	99%	0.4 %
B	54 sec	3402 sec	52%	10.1%
C	31 sec	1066 sec	62%	7.3%
D	52 sec	3126 sec	20%	0.42%
E	61 sec	3720 sec	62%	0.5%
F	30 sec	1087 sec	66%	0.49%
ours	31 sec	118 sec	0.35%	0.022%

首先，我們以第二類資料集評比各種方法在檢測大圖與子圖關係的成效。表二的結果顯示，ours在MR與IMR都明顯優於其他方法，且就On-Time時間來看ours比方法B~F來的快速，這是因為ours是以群母圖來比對，而非暴力法比對(C_2^n)；方法A檢測時間雖低於ours，但對於大圖與子圖關係毫無辨別能力。

表三、轉換色調之大圖與子圖比對結果[3]

	Off-Time	On-Time	MR	IMR
A	5.017 sec	0.096 sec	99%	0.06 %
B	57 sec	3693 sec	51%	10.3%
C	32 sec	1080 sec	62%	7.6%
D	58 sec	2781 sec	19%	0.41%
E	58 sec	3461 sec	61%	0.5%
F	34 sec	1137 sec	67%	0.53%
ours	30 sec	100 sec	1.05%	0.007%

表三為隨機選擇圖片轉換色調之大圖與子圖關係檢測結果，跟表一相比，色調的轉換對所有方法來說都稍稍影響了MR、IMR，但還是非常具有分辨能力。

6.3 參數敏感性測試

Ours採用aHash來判別關鍵截取圖塊是否為相似，表四呈現ours在不同aHash距離門檻下的表現，其結果顯示隨著aHash距離門檻變高，MR表現不一定，IMR卻會

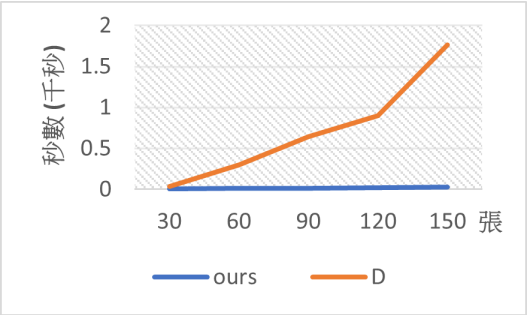
[1] off-time：為單一圖片處理時間，如尋找關鍵點等。
[2] on-time：為照片配對照片相似度檢測之處理時間。
[3] 色調轉換方法：
Step1. RGB色道平均值各乘1.3，計算出新的gray平均值(ave_r+ave_g+ave_b)/3，再計算RGB色道的增益係數。例如，R增益係數為gray_ave/ave_r。
Step2. 算出增益係數後，將RGB乘以各自的增益係數，超過255以255記。

升高，容易把不同群圖片合併於一群。

表四、ours對不同aHash distance比對結果

	Off-Time		On-Time		MR	IMR
distance < 10	29	sec	82	sec	0%	0%
distance < 12	30	sec	100	sec	1.05%	0.007%
distance < 14	30	sec	96	sec	0.70%	0.059%

最後，圖十五顯示Ours與D在不同數量圖片下的On-Time時間，可以看見以群母圖的概念能非常有效節省檢測時間。



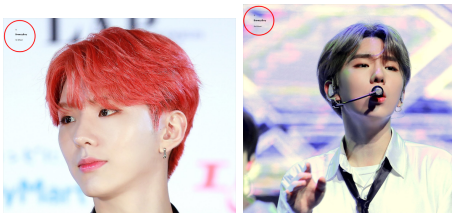
6.4 Case study

服裝與背景相似容易使得整體圖片與色階乍看相似(如圖十六)，使得aHash演算法較難分辨其為不同群的照片，但人物與動作不同能使得我們的方法在取關鍵點三角的步驟中出現區別，所以能判別其為不同組的照片。同理，在動作相似人物不同的情況(如圖十七)也會使得兩圖在取關鍵點三角的步驟中出現區別，所以依舊能判別其為不同組的照片。

我們的方法因採用SIFT取關鍵點，而SIFT的缺點可能在兩圖出現完全相同的浮水印時顯現(如圖十八)，在不同圖片中出現相同浮水印時使得SIFT抓關鍵點步驟時有一定機率被混淆，進而在aHash比對時有低機率被視為相似圖片。

圖十六、服裝背景相似，人物動作不同

圖十七、動作相似，人物不同



圖十八、左上角紅圈處出現相同浮水印

七、結語

這次的專題內容有用到對圖片的操作，是我們第一次接觸到的範圍，此專題一開始進行時並不是很順利，在熟悉的階段就花了不少時間，最後做出想、來的目標也跟一開始有落差，原要做手機上的APP，但後來考慮到手機可能無法負擔目前程式的運算，於是將目標改為電腦上的應用程式。未來希望還能再使鏡像、旋轉過的照片，都可順利檢測出來，前端使用者介面也加入讓使用者及時移動或刪除的選項，讓程式功能與使用者體驗更上一層樓。

八、銘謝

感謝指導教授在這一年來從我們題目的發想到專題實作的過程中給予我們許多寶貴的建議與方向，讓我們能及早發現並修正問題，幫助我們建立基礎概念，使我們的軟體顯得更加完善，才有了我們今天的專題成果。

參考文獻

[1]TESTING DIFFERENT IMAGE HASH FUNCTIONS. 2022年5月24日,取自 https://content-blockchain.org/research/testing-different-image-hash-functions/
[2]D. G. Lowe,Distinctive Image Features from Scale-Invariant Keypoints, International Journal of Computer Vision, 2004. 2022年5月24日,取自 https://www.cs.ubc.ca/~lowe/papers/ijcv04.pdf
[3]App front-end interface creation. 2022年5月24日,取自 https://www.wongwonggoods.com/python/pqt5-1/
[4]Python OpenCV. 2022年5月24日,取自 https://docs.opencv.org/4.x/d6/d00/tutorial_py_root.html