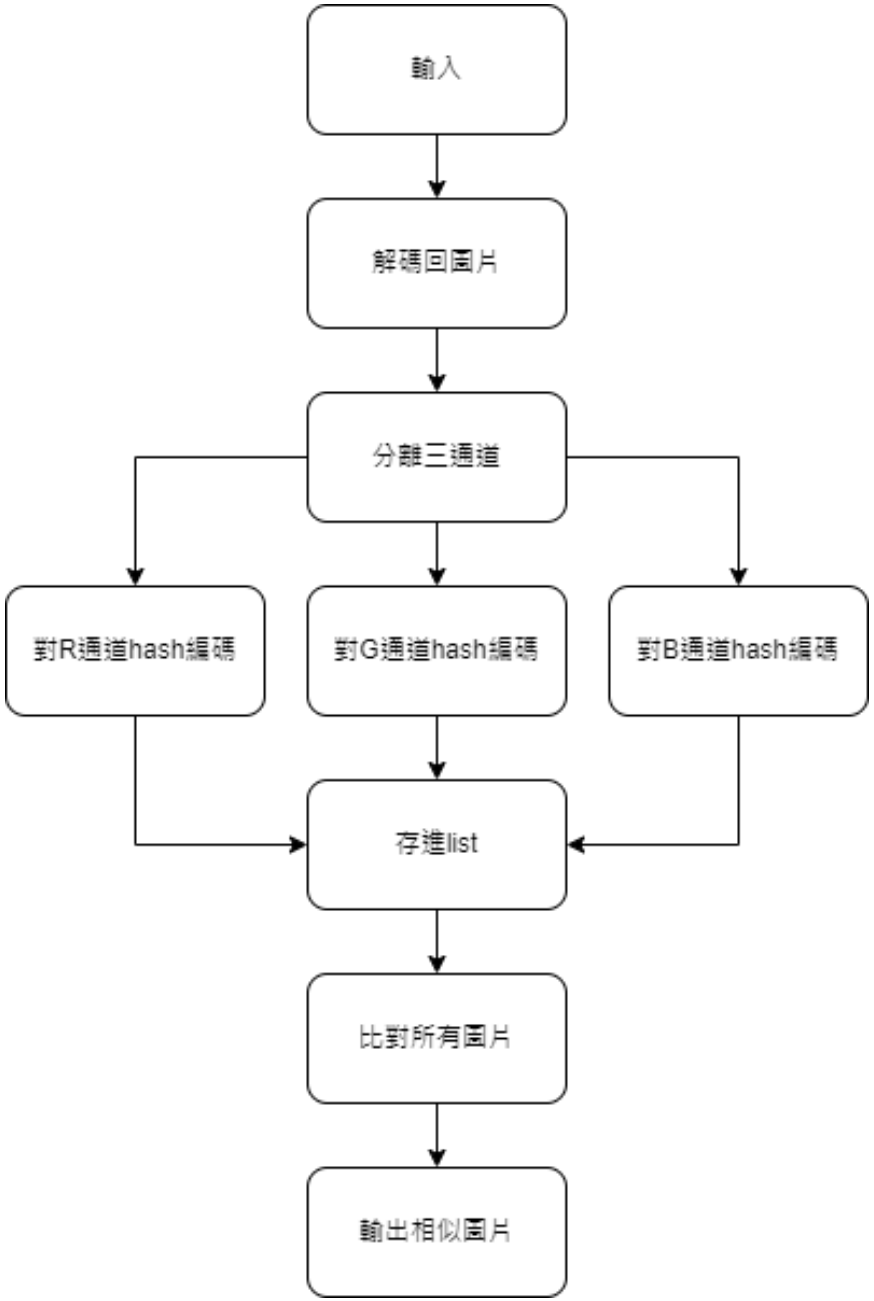
BLUR HASH

檢測相似圖片

2022.9.2



流程

components

主要分成三步驟，首先將hash code轉成bit map 再將bit map分割出三通道並對三通道進行hash 編碼，完成後存進list，最後再將每張圖片的hash value做比對找出相似圖片。

DECODE

細部介紹 - 一

```
for i in photo_list:
    img = np.array(blurhash.decode(i['blurhash'], 32, 32)).astype('uint8')[:,:,::-1]
    Hash_list.append(pHash(img))
```

將輸入的hash code轉成bit map，再將RGB三通道的順序對調成BGR(opencv圖片格式)，後存成np.array。

```python
def pHash(image):
    #image = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
    hash = []
    hash.append([])
    hash.append([])
    hash.append([])
    for k in range(3):
        imageSC = image[:,:,k]
        #dct = cv2.dct(np.float32(imageSC))
        dct = np.float32(imageSC)
        dct_roi = dct[0:,0:]
        #print(dct_roi.shape[0])
        avreage = np.mean(dct_roi)
        for i in range(dct_roi.shape[0]):
            for j in range(dct_roi.shape[1]):
                if dct_roi[i,j] > avreage:
                    hash[k].append(1)
                else:
                    hash[k].append(0)
    return hash
```

HASH ENCODE

細部介紹-二

採用基本的hash演算法，超過平均值的像素設為1其餘設為0，註解部分是dct hash演算法目前測試結果與基本hash各有優缺但速度較慢，也嘗試過直方圖比對結果較差。

COMPARE

細部介紹-三

```
for i in range(len(Hash_list)-1):
    for j in range(i+1, len(Hash_list)):
        ansB = cmpHash(Hash_list[i][0], Hash_list[j][0])
        ansG = cmpHash(Hash_list[i][1], Hash_list[j][1])
        ansR = cmpHash(Hash_list[i][2], Hash_list[j][2])
        if ansB < threshold and ansG < threshold and ansR < threshold:
            print(ansB, end = ' ')
            print(ansG, end = ' ')
            print(ansR, end = ' ')
            print(photo_list[i]['blurhash'] + ' ' + photo_list[j]['blurhash'])
```

目前採用對所有圖片一對一做比較，三通道均小於閥值者判斷為相似，時間複雜度為n^2效率較差但較不易有遺漏，未來可調整比對的方式。

圖片相似度方法比較

	直方圖	Hash(目前)	phash	Code cmp
速度	快	中等	略慢	極快
準確度	低	中	中	待驗證
可行性	高	高	高	待研究

優秀範例(皆為相異圖片)

失敗範例

好像只能說… 真的很像