ADIP 2022 專題報告

作者一顏郁芩[a]、作者二謝瑞榆[b]、作者三葉雨婷[c]
學號：110368155　實驗室：綜科 107　指導教授：高立人[a]
學號：110618053　實驗室：綜科 525　指導教授：李仕宇[b]
學號：110368404　實驗室：綜科 205　指導教授：黃育賢[c]

摘要 — 本算法針對一組亂序的可能存在有雜訊、亮度過亮過暗和濾鏡處理等問題的影像幀進行處理並排序。算法使用均值濾波、平均亮度調整等方法進行影像幀的處理，並使用 SIFT 尋找圖像幀的特徵點，通過特徵點距離進行排序。算法在多個數據集進行運行時間、SRCC 和平均 MSE 評估分數的性能評估，其中，平均運行時間為 395.2 秒，平均 SRCC 為 0.3846，平均 MSE 為 2622.02。

簡介

本次算法要處理一組亂序的圖像，圖像可能存在雜訊、亮度過亮過暗及經過濾鏡處理等問題，並對其進行排序。為了對影像幀圖像進行正確的排序，算法對圖像組進行預處理，其中包括去雜訊、亮度調整等。

常見的去雜訊算法分為空間域去噪算法和變換域去噪算法。其中，均值濾波是用像素鄰域的平均灰度來代替像素值，使用於脈衝噪聲，因為脈衝噪聲的灰度級一般與周圍像素的灰度級不相關，而且亮度高出其他像素許多。通過對數據集的觀察，圖像可能存在的雜訊為椒鹽雜訊和彩色點雜訊。而中值濾波是一種非線性濾波，在處理脈衝噪聲以及椒鹽噪聲時效果極佳，能夠有效的保護好圖像的邊緣信息。因此，算法採用中值濾波的空間域去雜訊方法進行圖像的去雜訊。空間域去噪算法是通過分析在一定的窗口內，中心像素與其他相鄰像素之間在像素空間的直接聯繫，來獲取新的中心像素值的方法。

對於圖像組的亮度不均問題，即存在某些圖像亮度過高，某些圖像亮度過低的現象，算法採用平均亮度調整的方法進行處理。視頻幀的圖像預處理應保證圖像間的亮度保持一致或相似，避免出現亮度範圍差異過大而導致的視頻“閃爍”現象。因此，算法考慮圖像間的亮度範圍，將圖像的亮度調整到平均亮度範圍。出於對異常亮度的數據數量的考慮，算法考慮平均亮度的方式，更有利於得到數據集得到更合適的亮度，使得圖像間的亮度範圍保持相似，確保相鄰圖像不會出現突兀的亮度變化。

對於亂序圖像間的排序問題，一種有效的衡量圖像間距離或者相似度的方法會對最終的準確性起到關鍵作用。為了進行準確的排序，算法的排序部分將排序問題分解為兩個子問題：（1）尋找首幀（2）準確計算圖像間的距離，尋找最近距離的圖像。對於首幀問題，我們認為對所有圖像進行相似度計算，尋找每幀圖像最高相似度的兩張圖像，則當前圖像與其相鄰的圖像的相似度會很高，而對於不相鄰的圖像，則相似度會比較低。那麼，第一幀和最後一幀圖像的僅有一張相鄰的圖像，即與首幀和尾幀保持高相似度的圖像也僅有一張。對於第二個問題，算法使用 SIFT 尋找圖像間的特征點，並將特征點的距離作為衡量圖像間距離的方法。即，在確定首幀的情況下，算法將特征點距離近的圖像認為是相鄰圖像，進行排序。

提出方法

本實驗方法分為兩個部分，一是去除雜訊，二是將影像排序。針對被干擾的視頻幀，本算法提出一個可以自動判斷是否需進行處理的決策條件，並依照其結果進行下一步的處理。第一種影片的幀被填入黑白椒鹽及彩色雜訊的情況，利用計算圖片像素出現次數的方式[1]，由於雜訊的出現會造成各個顏色通道像素的極值，也就是 0 和 255 的出現次數高於中間，如圖 1 所示。基於上述特徵，本算法提出的分類雜訊的條件為，將影像中藍色通道值為 0 出現的次數為呈現最高之影像判斷為帶有雜訊，並將其影像進行下一步去噪處理。本實驗方法採用中值濾波[1]的空間域為影像去除雜訊。透過取卷積核 3*3 範圍當中所覆蓋像素中的中值作為錨點的像素值，其結果能有效的去除視頻幀的黑白椒鹽，去除雜訊的結果與直方圖如圖 2 所示。

對於亮部不均及被添加了濾鏡造成色調呈現單一的視頻影像。首先對受干擾的影像數據進行觀察，若影像過暗三個顏色通道的值皆會偏低，相反的過亮的影像會有較高分佈的像素值。若影像呈現單一色調的濾鏡效果，會讓該色調的像素特別突出，而其餘兩個顏色通道的值會被降低。基於上述原因本算法透過計算出視頻三個顏色通道的平均值。並利用平均值與每張圖片的三通道顏色進行比較，同時計算該圖片的該顏色通道與平均值相差數值。其相差值將會被採用來調整通道整體的數值，讓原本過低的顏色通道加上缺少的數值，過高的通道會減去多餘的數值[1]，藉此讓紅、綠、藍三色的像素分佈皆保持在合理的平均範圍。此方法不僅調整了視頻中亮度不均的影像，也將原本只有單一色調的影像恢復為原本的顏色分佈。此作法下之彩色圖像過亮之調整如圖 3、過暗調整如圖 4，而濾鏡調整之結果如圖 5、圖 6 所示。

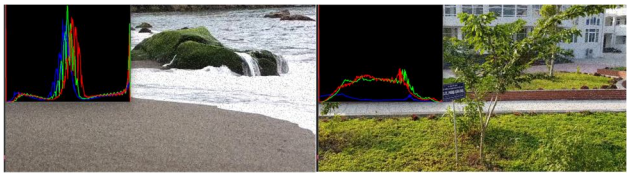

圖 1 加入椒鹽的受干擾影像及其直方圖

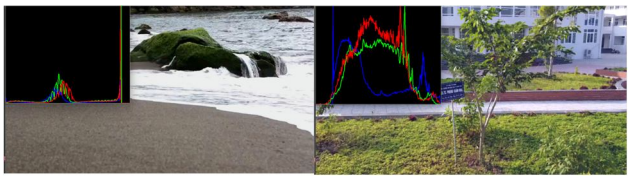

圖 2 利用自動偵測找出並去噪處理之後的結果

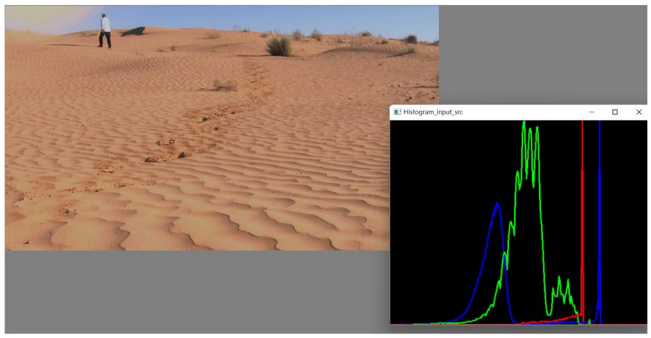

圖 3 利用平均值比對自動調整過亮影像結果

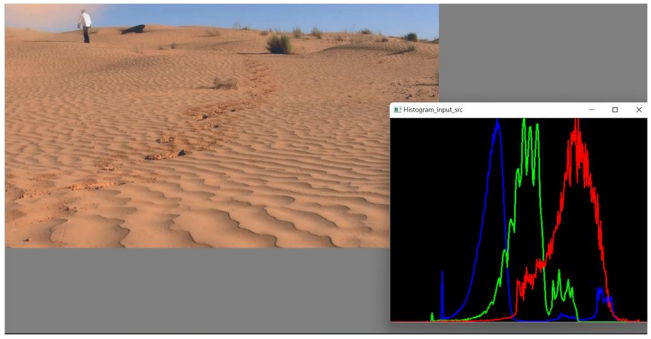

圖 4 利用平均值比對自動調整過暗影像結果

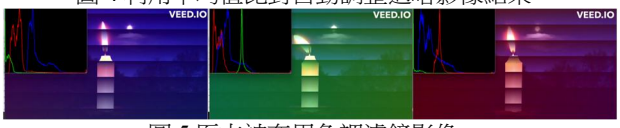

圖 5 原本被套用色調濾鏡影像

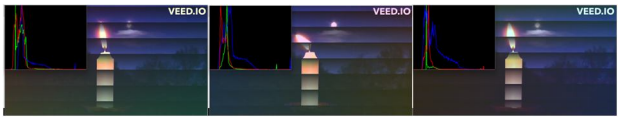

圖 6 利用平均值比對自動調整色調濾鏡結果影像

排序部分輸入為已經去雜訊處理之影像，此步驟又可細分為兩部分，一部分是找初始幀，另一部分則是剩下幀的排序，輸出則為排列好之影像編號 txt 檔。初始幀的尋找使用了 opencv 作 PSNR 計算[1]與相似度比較之概念：除第一幀和最後一幀之外，中間的幀數應有兩幀與自己相近，其 PSNR 間的差值亦會較小。如圖 7 所示。以此概念作判斷，|a1-a2|會大於 |b1-b2|，故全部比較過後取差值最大的幀即為第一幀或最後一幀。

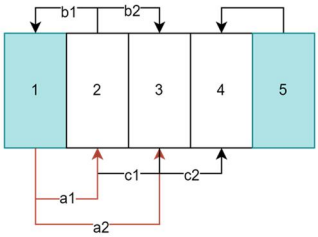

圖 7 首末幀判斷示意圖

其餘幀的排序使用到了 SIFT[6]特徵點偵測方法以及特徵點在不同圖片中的移動距離來判斷，取平均移動距離最小之幀做為下一幀。SIFT 找尋之特徵以 blob 為主，corner 為輔，雖能廣泛應用於電腦視覺領域但有計算量大以及速度慢的情形。但基於參考資料[5]中與其他演算法的比較結果，我們依然選用 SIFT 作為此報告主要之排序演算法。下圖 8 為數據集中其中一張圖片套用 SIFT 演算法後之結果。在此步驟會遇到的問題是絕大多數的數據集中的資料使用 SIFT 演算法之後特徵點的數量單張都會在數千個，加上使用之平均距離計算會使用到大量的運算故希望盡量減少特徵點數量。

圖 8 套用 SIFT 演算法之特徵點偵測結果

採用高斯模糊[7]以及線性 resize[8]來降低解析度與減少特徵點。以每一數據集第一張照片之特徵點數量作為判斷基準，判斷流程如圖 9 所示，依據第一張照片中特徵點數量進行模糊和降低解析度。經由 matcher 所進行特徵點配對示意如圖 10 所示，採用 opencv 的 matcher[9]來進行特徵點配對，找尋 matcher 配對的每一對特徵點在各自圖片上的座標並進行距離計算，整張圖計算完再算平均。

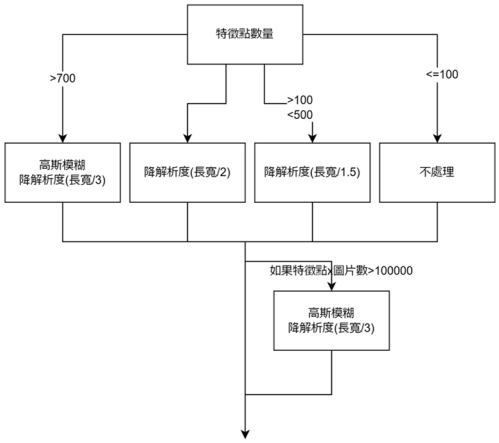

圖 9 模糊或降低解析度流程圖

圖 10 利用 matcher 進行特徵點配對後繪製結果

實驗結果與比較

為了測試算法的效果，我們在 15 種不同的數據集上進行處理與排序。數據集從不同類型的視頻中進行幀的提取，並進行不同類型的破壞，如加入雜訊、調亮或調暗圖像的亮度、進行濾鏡處理等。

算法通過運行時間、準確度、評估分數等方式進行效果的衡量。其中評估的分數包括 Spearman's rank correlation coefficient（SRCC）和平均 MSE。對於 SRCC，可以用於表示算法排序結果和正確排序結果的相關性，其計算公式如式 1 所

示。算法分別計算從排序結果的 SRCC 值和相反的排序結果與正確的排序結果的 SRCC，並且取最大值作為算法的 SRCC 值，

$$\mathrm{SRCC}=\left|1-\frac{6\sum \mathrm{d}_i^2}{n\left(n^2-1\right)}\right| \quad (1)$$

其中，d^i 代表兩幀影像排序的差，n 則為該部影片總幀數。越接近 1 則其正確率越高。

對於平均 MSE，我們將它應用在比較重新拼接後的幀排序圖像與 Ground truth 原始幀順序的差異，並在統計完整影片後取平均。其計算公式如式 2 所示，越接近零代表其正確率越高。

$$\mathrm{MSE}_{\mathrm{Avg}}=\frac{1}{\mathrm{N}}\sum_{\mathrm{x}=1}^{\mathrm{N}-1}\frac{1}{mn}\sum_{i=0}^{m-1}\sum_{j=0}^{n-1}\left[I_x(i,j)-K_x(i,j)\right]^2 \quad (2)$$

正確率的算法為和正確排序的 txt 檔案做比較，若排序以及數字均正確者才認為是正確。

我們分別計算算法對 15 個數據集的測試結果，並且計算 15 個數據集的平均結果。如表 1 所示。對於不同數據集，圖像存在的場景和問題各不相同，去雜訊和排序的難度也不一致。從對各個數據集的平均結果看，我們的算法的平均運行時間是 395.2 秒，平均的正確率為 0，平均的 SRCC 和 MSE 分別為 0.3846 和 2622.02。

因在找尋第一幀的部分就不一定正確故所有數據集的正確率均為零，但有部分幀有正確連續排列。整體效能可從 SRCC 以及 $\mathrm{MSE}_{\mathrm{Average}}$ 做輔助判定。

表 1 算法在 15 個數據集的測試效果

影像序列	執行時間（s）	正確率	SRCC	$\mathrm{MSE}_{\mathrm{Avg}}$
Beach	298	0	0.398	3194.29
Boat_style	1074	0	0.551	1664.39
Candle_style	631	0	0.070	794.981
CCTV	327	0	0.221	454.64
Costline	247	0	0.747	3983.49
Desert	204	0	0.368	3747.55
DMC	720	0	0.282	1074.4
Flyout	337	0	0.874	1261.69
Helltaker	272	0	0.259	735.767
PUBG	83	0	0.445	6428.86
RushPixar	427	0	0.004	5189.52
School	260	0	0.522	4105.34
Soccer_style	152	0	0.195	1540.7
TKUC	260	0	0.462	4443.74
Typing	684	0	0.371	670.437
平均	395.2	0	0.3846	2622.02

結論

設計出的演算法期望能將被打散的幀排序回原來的影片，依結果來看雖然部分資料集尚未達到 50%以上的正確率，但在部分的資料集上具有尚可的結果。另外對於數據集各自的雜訊處理，此演算法可將具有色偏的彩色圖片顏色統一，或者是將椒鹽雜訊清除掉。但針對被摻入過度密集雜訊的影像，其處理的效果有限，並可能會影響到後續的排序的結果。排序部分未來可利用改變特徵點偵測方式，例如使用 FAST 取代 SIFT，以及利用 GPU 加速等等來加速此演算法。另外在尋找第一幀的部分，可利用不指定第一幀而是從任意幀往外延伸出去之方式來提高排序正確率。

參考資料

[1] https://docs.opencv.org/3.4/d8/dbc/tutorial_histogram_calculation.html OpenCV: Histogram Calculation

[2] https://blog.csdn.net/tengfei461807914/article/details/83626123 均值濾波&高斯濾波&中值濾波

[3] https://docs.opencv.org/3.4/dd/d4d/tutorial_js_image_arithmetics.html OpenCV: Arithmetic Operations on Images

[4] https://www.cnblogs.com/theodoric008/p/9288635.html C++ opencv 計算兩張圖像的 PSNR 相似度

[5] https://chtseng.wordpress.com/2017/05/22/%E5%9C%96%E5%83%8F%E7%89%B9%E5%BE%B5%E6%AF%94%E5%B0%8D%E4%BA%8C-%E7%89%B9%E5%BE%B5%E9%BB%9E%E6%8F%8F%E8%BF%B0%E5%8F%8A%E6%AF%94%E5%B0%8D/圖像特徵比對(二)-特徵點描述及比對

[6] https://physics.nyu.edu/grierlab/manuals/opencv/classcv_1_1SiftFeatureDetector.html cv::SiftFeatureDetector Class Reference

[7] https://www.opencv-srf.com/2018/03/gaussian-blur.html Gaussian Blur OpenCV Tutorial C++

[8] https://learnopencv.com/image-resizing-with-opencv/ Image Resizing with OpenCV

[9] https://docs.opencv.org/3.4/d5/d6f/tutorial_feature_flann_matcher.html Feature Matching with FLANN