Image Segmentation

何景盛 B08901123

2023/11/30

Abstract

影像分割（Image Segmentation）是電腦視覺領域中的一個重要任務，是指將數位影像劃分成具有相似特徵或屬性的區域。而影像分割通常被視為許多影像應用中重要的前處理步驟，分割出具有義意的特徵區域，從而以更精確地理解和處理影像內容。此技術在許多應用領域中都發揮了重要作用，包括醫學影像分析、自動駕駛、物體檢測等。此專題研究為實作傳統數位影像分割技術並理解其意義。

1 Introduction

此專題研究專注於研究影像分割，並運用多種技術，如 Fast Segment[2]、K-means、multi-stage merge、Watershed 和 Entropy Rate Superpixel(ERS)[3]，以實現對數位影像的劃分。通過運用超像素的多樣性特徵，包括顏色差、亮度差、邊界梯度、材質、面積等資訊，進行影像分割。

1. Fast Segment：是一種區域成長的影像分割方法，每個 pixels 與相鄰的 pixel 區域差，根據 threshold 進行合併或建立新區域。
2. K-means：隨機建立數個中心點，每個 pixels 根據與中心點距離差和 ycbcr 差進行分區。
3. multi-stage merge：用於過分割或相似區域被分割時，進行合併，不同層可對不同的資訊進而決定合併條件。
4. Watershed：基於數學形態學的分割方法，將圖像視為地形地勢圖，從梯度最小開始提升水位，形成不同區域。
5. Entropy Rate Superpixel(ERS)：基於 Entropy Rate 與 graph 進行圖像分割。

2 Fast Segment

Fast Segment 的主要想法是運用圖片中每個 pixels 的強度作合併。假設一張圖片是 M*N 的大小，而 C[m, n] $(1 \leq m \leq M, 1 \leq n \leq N)$ 則是 pixels 的強度，一開會設定兩個值用作為算法的判斷，$threshold$ 為檢查 pixels 之間的強度差，Δ 為最後處理區域過小的問題用。

這個算法會從圖片的 C[1, 1] 開始往右掃，直到第一行完成，再到第二行，如此類推，直到整張圖片完成。而在掃的過程也會進行分區，例如在一開始需要初始化設定 C[1, 1] 為 $region1$，接著到 C[1, 2]，這時會檢查它跟左邊的 pixel 的差，如果小於一個 $threshold$，就會把 C[1, 2] 合併在 C[1, 1] 的區域，並計算 $region1$ 裡 pixels 的平均值，大於 $threshold$ 則開一個新區域為 region2，如此類推，直到第一行完成。到了 C[2, 1] 則是檢查跟上面的 pixel 差，小於 $threshold$ 就合併算平均，大於則開新區域。到了 C[2, 2] 就有點不同，需要檢查上面跟左邊的 pixels 差，如果跟上面差小於 $threshold$ 就跟上面合併，跟左邊差小於 $threshold$ 就跟左邊合併，兩者都大於 $threshold$ 則開新區域，兩者都小於 $threshold$ 則把上面區域的 pixels 與 C[2, 2] 合併到左邊的區域裡，並重新算平均。如此類推向上向左檢查 pixels 差，直到整張圖都掃一邊，最後再看有沒有一個區域內的 pixels 數量是小於 Δ 值，如果有則把合併到鄰近的區域裡。最後把得到的區域依 pixels 數量進行排序。

詳細流程和算法可參考 Fast Segment[2] 這篇論文中。

3 K means

在使用 K means 之前，可以先把圖片從 rgb 換成 ycbcr 取得其強度和色度，以下 Eq.1為其轉換式子：

$$\begin{bmatrix} y[m,n] \\ cb[m,n] \\ cr[m,n] \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.331 & 0.5 \\ 0.5 & -0.419 & -0.081 \end{bmatrix} \begin{bmatrix} r[m,n] \\ g[m,n] \\ b[m,n] \end{bmatrix} \tag{1}$$

以下將解釋如何使用 K means 在影像分割的流程：

1. 在初始化時，隨機選擇 K 個中心點。以 (m_k, n_k) 表示中心，以 (y_k, cb_k, cr_k) 表示顏色，其中 $k = 1, 2, ..., K$。為了中心點合併效果更好，可以用 sobel filter 來計算圖片的梯度，並把 K 個中心點移到附近梯度較小的點，以至中心點較為平滑。

2. 對於影像中每個 pixels 與初始化中 K 個中心點運用位置差和 ycbcr 差來計算距離，以下 Eq.2為計算距離之式子，當中 λ_1 和 λ_2 是可以自行調整，作為位置差與強度差之權重。

$$d[m,n,k] = \sqrt{\lambda_1[(m-m_k)^2 + (n-n_k)^2] + \lambda_2(y[m,n]-y_k)^2 + (cb[m,n]-cb_k)^2 + (cr[m,n]-cr_k)^2} \quad for \quad k = 1, 2, ..., K \tag{2}$$

3. 如果 $d[m,n,h] < d[m,n,k] \quad for\ all\ k \neq h$ 就將像素 [m, n] 分配給第 h 的中心點形成區域。

4. 對於每個區域重新計算 (m_k, n_k) 區域中心點和 (y_k, cb_k, cr_k) 區域平均 ycbcr，如下 Eq.3。

$$\begin{aligned} m_k &= \underset{[m,n] \in k^{th} region}{\text{mean}} m \\ n_k &= \underset{[m,n] \in k^{th} region}{\text{mean}} n \\ y_k &= \underset{[m,n] \in k^{th} region}{\text{mean}} y[m,n] \\ cb_k &= \underset{[m,n] \in k^{th} region}{\text{mean}} cb[m,n] \\ cr_k &= \underset{[m,n] \in k^{th} region}{\text{mean}} cr[m,n] \end{aligned} \tag{3}$$

5. 重复步骤 2-4 到某個选遞次數，最後得出分割區域。

4 Multi-stage merge

Multi-stage merge 由多層 merge 組成，在每層 merge 會對不同的資訊敏感度作出不同的合併，當中會根據 *region A* 和 *region B* 的位置差、顏色差、亮度差、邊界和材質，使用此資訊來計算距離，並用一個 threshold 來判斷是否需合併。

位置差、顏色差和亮度差與 Kmeans 中 Eq.2方法相似。

邊界則是使用 sobel 和 laplacian 對整張圖進行梯度運算，並計算區域之間的邊界平均梯度。

$$G_{border} = \underset{(m,n) \in regionA\ and\ regionB\ edge}{\text{mean}} G, \quad L_{border} = \underset{(m,n) \in regionA\ and\ regionB\ edge}{\text{mean}} L$$

1. Sobel filter: 基於梯度的濾波器，用於檢測圖像中的邊緣。它利用卷積操作計算每個像素點的梯度，其算式如下，當中 A 為影像。

$$G = \sqrt{G_x^2 + G_y^2}, \quad G_x = \frac{1}{4}\begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} * A, \quad G_y = \frac{1}{4}\begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} * A$$

2. Laplacian filter: 用於檢測圖像中的區域性亮度變化，強調了圖像中的高頻細節，其算式如下。

$$L = \frac{1}{8}\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} * A$$

材質則是根據區域中的平均梯度，以下為其運算式。

$$texture_x = (mean|G_x|)^\alpha, \quad texture_y = (mean|G_y|)^\alpha \quad where\ \alpha < 1$$

區域之間距離差之式子如下 Eq.4

$$\begin{aligned} distance &= \sqrt{\lambda_1\Delta_{position} + \lambda_2\Delta_{intensity} + \Delta_{color} + \lambda_3 G_{border} + \lambda_4 L_{border} + \lambda_5\Delta_{texture}} \\ &where \\ &\Delta_{position} = (m_A - m_B)^2 + (n_A - n_B)^2, \\ &\Delta_{intensity} = (y_A - y_B)^2 \\ &\Delta_{color} = (cb_A - cb_B)^2 + (cr_A - cr_B)^2 \\ &\Delta_{texture} = (texture_{x,A} - texture_{x,B})^2 + (texture_{y,A} - texture_{y,B})^2 \end{aligned} \tag{4}$$

進行合併的條件如下：
如果 $distance < threshold$，則把兩個區域合併，$distance \geq threshold$，則不合併。

在 Multi-stage merge 中，可以根據調節每層 merge 中 Eq.4的 λ_1、λ_2、λ_3、λ_4 和 λ_5 的參數，從而改變不同資訊的權重，促使改變區域合併的條件。例如在第一層設 λ_1 較大，這樣可以更嚴格地處理位置相差較遠的區域不能合併。使每一層都可處理不同的資訊。

5 Watershed

Watershed 此方法更適用於灰度圖像，因此會先把原本 RGB 的圖轉換成 ycbcr，再取得當中的強度 y 來進行分割。Watershed 的基本概念是區域的中心應該具有較小的梯度，而區域的邊界應該具有較大的梯度，基於這性質進而衍生出其算法，但通常會有過分割的狀態。因此在 Watershed 分割後，需進行合併，減少過分割的情況，在此用 multi-stage merge 來完成。

以下將解釋如何使用 Watershed 在影像分割的流程：

1. 使用 sobel 來計算影像的梯度 G，沿著 x 軸和 y 軸執行邊緣檢測，分別獲得 G_x 和 G_y。然後計算 $G[m,n] = \sqrt{G_x[m,n]^2 + G_y[m,n]^2}$
2. 將 $G[m,n]$ 進行量化為幾個級別成 $L[m,n] = round(G[m,n]/Q)$。以下圖 Fig.1為已被量化作例子進行解釋。

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 1: L

3. 一開始先對 $level = 0$ 的情況執行二值分割 $(L == 0)$，可得下圖 Fig.2有三個區域

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 2: Level 0

4. 再漸漸增加 L 等級判定，對 $level = level + 1$ 進行二值分割與分配區域。

5. 為滿足 $L[m,n] = level$ 的 pixels 分配區域可分為下面三類：

   - 如果 $L[m,n] = level$ 且與某些現有區域相鄰，則將其分類為該區域。

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 3: Level 1 one adjacent region

   - 如果某些 $L[m,n] = level$ 的 pixel 鄰近兩個或更多區域，我們可以自行分配優先順序，例如根據方向的優先順序先上、再是下、接著是左、最後是右。

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 4: Level 1 more than one adjacent region

   - 如果 $L[m,n] = level$ 的 pixel 尚未被前兩個情況分配，則對所有 piexls 重複前兩個情況的步驟直到沒有 $L[m,n] = level$ 的 pixel 可以分配區域。

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 5: Level 1 no adjacent region

6. 如果某些 $L[m,n] = level$ 的 pixels 無法分配到任何區域，可以將它們視為新的區域。

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 6: Level 1 assign new region

7. 重複執行步驟 4、5 和 6。直到影像中的所有 pixels 都可以分配到區域。

$$L = \begin{bmatrix} 2 & 1 & 3 & 3 & 2 & 1 & 1 \\ 0 & 0 & 0 & 1 & 0 & 0 & 1 \\ 2 & 1 & 3 & 2 & 0 & 0 & 1 \\ 3 & 4 & 4 & 5 & 2 & 2 & 1 \\ 1 & 2 & 3 & 4 & 3 & 3 & 3 \\ 0 & 1 & 2 & 4 & 2 & 2 & 3 \\ 0 & 1 & 3 & 3 & 2 & 1 & 1 \end{bmatrix}$$

Figure 7: Result of Watershed method

6 Entropy Rate Superpixel Segmentation(ERS)

在 Entropy Rate Superpixel Segmentation 中，把先前 Watershed 的方法換成 ERS，再用 multi-stage merge 來完成分割與合併，令區域變得較大。ERS 影像分割演算法的核心思想是利用 graph $G = (V, E)$ 來表示影像，並透過 Entropy rate 隨機在 graph 上移動，尋找具有最大目標函數的邊，將其加入結果的 graph $G' = (V, A)$ 中，而 $A \subseteq E$。最終遍歷所有邊 E，得到圖 G' 為分割的結果。

詳細流程和算法可參考 Entropy Rate Superpixel Segmentation [3] 這篇論文中。

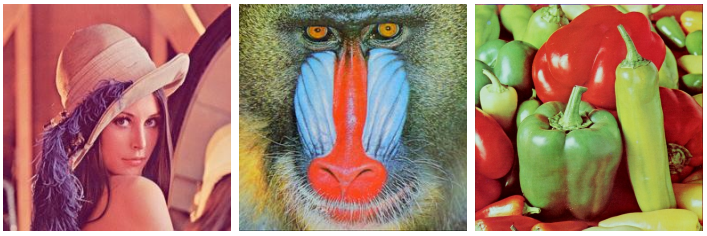
7 Experiments

7.1 Testing data and Ground truth

本次研究的測試圖片為 Fig.8(a) Lena、Fig.8(b) Baboon 和 Fig.8(c) Peppers。Ground truth 為 Fig.9(a) Lena、Fig.9(b) Baboon 和 Fig.9(c) Peppers。

評比分割的好壞以 Intersection over Union(IOU) 為基準，以下 Eq.5為 IOU 的式子。

$$\begin{aligned} IOU &= \sum_{n=1}^{N} w_n \max_m \frac{|A_n \bigcap B_n|}{|A_n \bigcup B_n|} \\ &where \\ &w_n : \frac{|A_n|}{|Image|},\ |\bullet| : the\ \#\ of\ region, \\ &A_n : n^{th}\ region\ in\ ground\ truth, \\ &B_m : m^{th}\ region\ by\ segmentation\ method \end{aligned} \tag{5}$$



(a) Lena (b) Baboon (c) Peppers

Figure 8: Testing images

(a) Lena (b) Baboon (c) Peppers

Figure 9: Ground truth images

7.2 Results and Discussion

在 Fig.10、Fig.11和 Fig.12為每一個分割算法的結果，每一個算法以前 16 多個 pixels 的區域作顯示，因此可以看到每個算法都有 16 張圖，白色的部分為分割區域。

IOU(%)	Lena	Baboon	Pepper	Average
Fast Segment	50.5	33	61.9	48.5
K-means	42.3	35.1	38.9	38.8
Watershed	58.5	47.5	69.6	58.5
ERS	62.1	48.7	64.7	58.5

Table 1: Result of IOU

Time consuming(s)	Lena	Baboon	Pepper	Average
Fast Segment	1.1179	1.1469	1.185	1.150
K-means	43.501	45.573	44.46	44.511
Watershed	2.522	2.786	2.992	2.767
ERS	2.881	2.854	3.089	2.941

Table 2: Result of time consuming

從結果可以觀察到整體 Watershed 和 ERS 在平均 IOU 和平均運算時間相約為佳，而 K-means 在分的區域會比較不完整及碎片化，並且在沒有平行化運算的用時較長，Fast Segment 在平均運算時間為最佳，因為核心算法只需把圖片走一遍便可得知分區。在影像上使用 Watershed 算法則會出現過分割，因此需要 multi-stage merge 來減少過分割的情況。

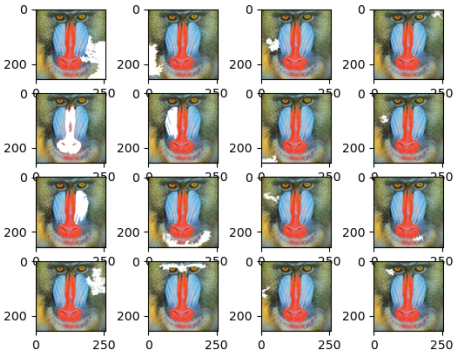
(a) Fast segment

(b) Kmeans

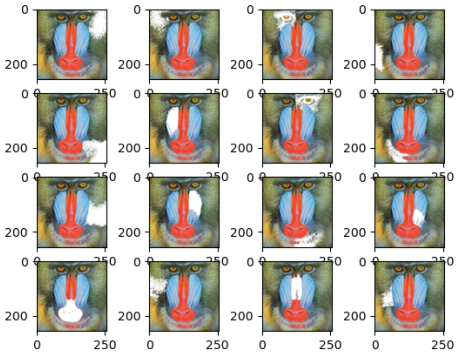
(c) Watershed

(d) ERS

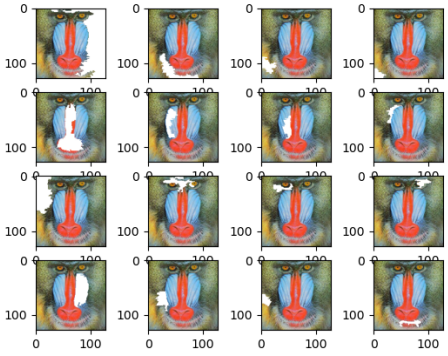
Figure 10: Lena



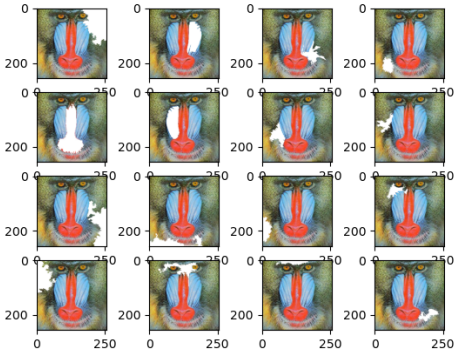
(a) Fast segment



(b) Kmeans

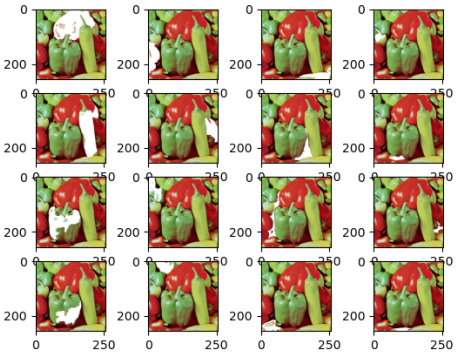


(c) Watershed

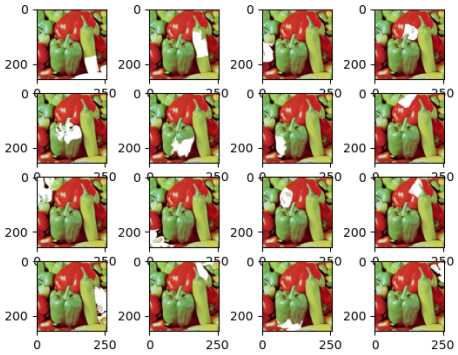


(d) ERS

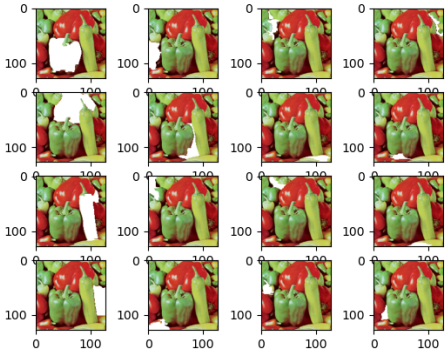
Figure 11: baboon



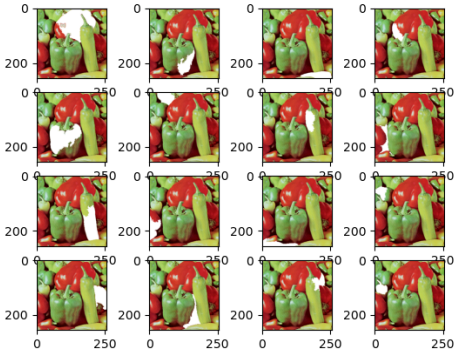
(a) Fast segment



(b) Kmeans



(c) Watershed



(d) ERS

Figure 12: peppers

References

[1] 專題資料

[2] http://djj.ee.ntu.edu.tw/FastSegment.pdf

[3] https://www.merl.com/publications/docs/TR2011-035.pdf