

Homework No.9

Student ID: 41047902S Name: 鄭淮薰

Problem Statement


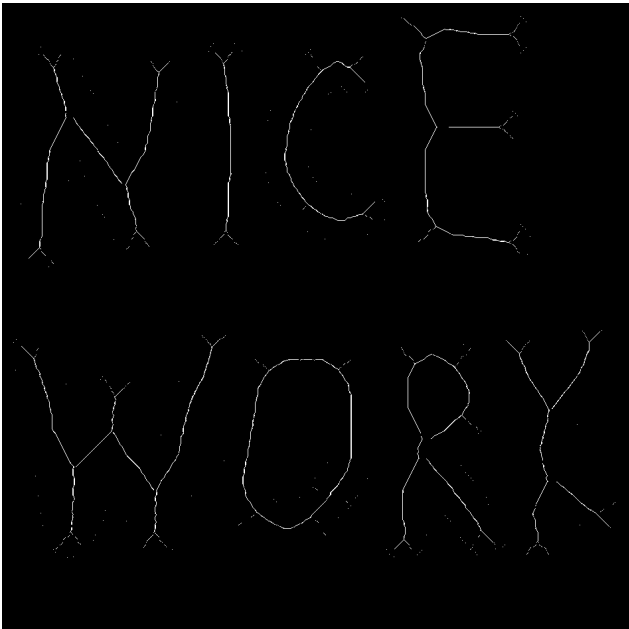
Implement the Lantuejoul’s skeletonization method using the structuring element B

Apply the method to the following input image :

Structuring Element B	Input Image
$\begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	

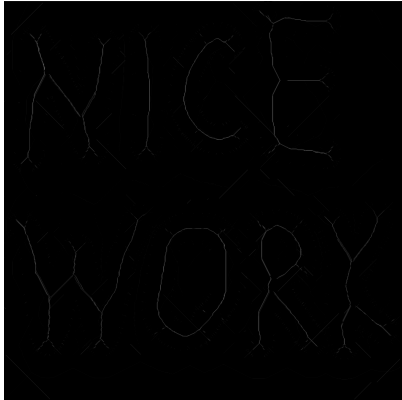
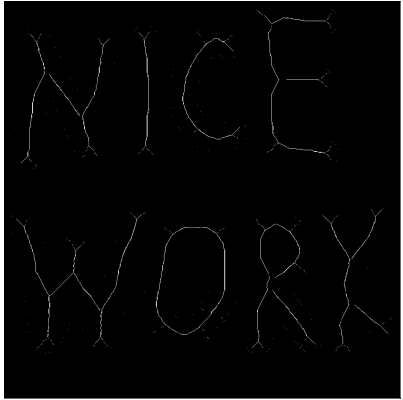
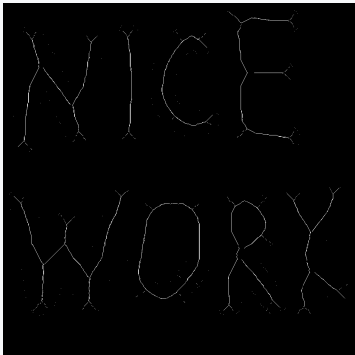
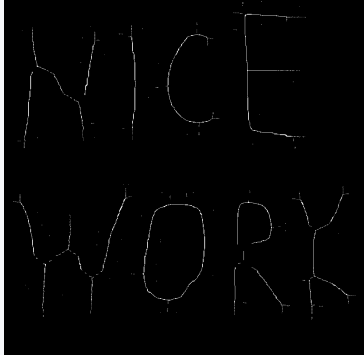
Results

Fig1.1是一張 `input.png` 作為 input，將其使用 structuring element B 做 Lantuejoul’s skeletonization method 後的結果則為 Fig1.2。

Fig1.1 Input Image	Fig1.2 Skeletonized Image
	

Comment

本次作業要實作 Lantuejoul's skeletonization 方法，首先我使用 cv2 套件將 `input.png` 讀入，並使用 np.array 建立 structuring element B。實作我使用 cv2 套件來做 erosion、dilation 及 subtract 等集合運算。過程中我發現若將圖像先做 thresholding 最後的成果會更好，這是因為 skeletonization 是一種基於邊緣的算法，即它需要提取圖像中的邊緣來生成 skeleton。而灰度圖像中的每個像素值都代表著該點的亮度或灰階程度，而不是明確的邊緣信息。為了獲取清晰的邊緣信息，必須將圖像轉換為二值圖像，其中只有黑色和白色兩種顏色表示圖像中的物體和背景。因此，從下表 Fig2.1、Fig2.2 兩張圖可發現，thresholding 可以清晰地標示出圖像中的物體輪廓和邊緣，使得其結果 Fig2.2 skeleton 更明顯。另外，我亦有嘗試使用不同的 structuring element 來實作，結果如 Fig3.1、Fig3.2 所示。

Fig2.1 No Tresholding	Fig2.2 Tresholding
	
Fig3.1 Structuring element B_1	Fig3.2 Structuring element B_2
$B_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 1 & 1 \\ 0 & 1 & 0 \end{bmatrix}$	$B_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$
	

Source Code

```
import cv2
import numpy as np

# Read image
img = cv2.imread('input.png', cv2.IMREAD_GRAYSCALE)

# Create structuring element
element = np.array([[0, 1, 0], [1, 1, 1], [0, 1, 0]], dtype=np.uint8)

# Apply Lantuejoul's skeletonization
_, img = cv2.threshold(img, 127, 255, 0)
size = np.size(img)
skel = np.zeros(img.shape, np.uint8)

while True:
    eroded = cv2.erode(img, element)
    temp = cv2.dilate(eroded, element)
    temp = cv2.subtract(img, temp)
    skel = cv2.bitwise_or(skel, temp)
    img = eroded.copy()

    zeros = size - cv2.countNonZero(img)
    if zeros == size:
        break

# save image
cv2.imwrite("skel.png", skel)
```