111511167 黃弘智

DIP Homework Chapter 9

1. Please recognize the image: text-broken.tif. Please describe your method, procedures, final text file and print out the source code?

There are three function blocks in Python code. Here are the settings:

Block name	parameter
Binary Thresholding(BT)	Threshold = 127, Type = THRESH_BINARY
Morph. Dilation (MD)	Kernel Size = 3x3, Iterations = 1, Op = Dilate
EasyOCR(OCR)	Language = 'en', Mode = paragraph

Path of Each Picture:

Path 1: Original Image Recognition (Control Group)

Input Image -> OCR -> Text Output

Path 2: Broken Text Restoration & Recognition (Experimental Group)

Input Image -> BT -> MD -> OCR -> Saved Image -> OCR -> Text Output

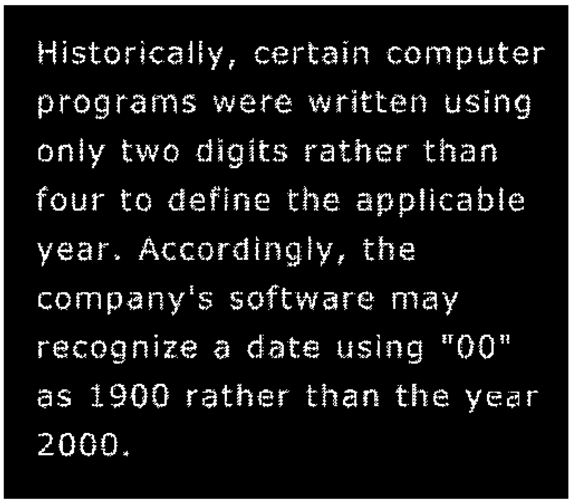

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Fig 1-a Original Picture

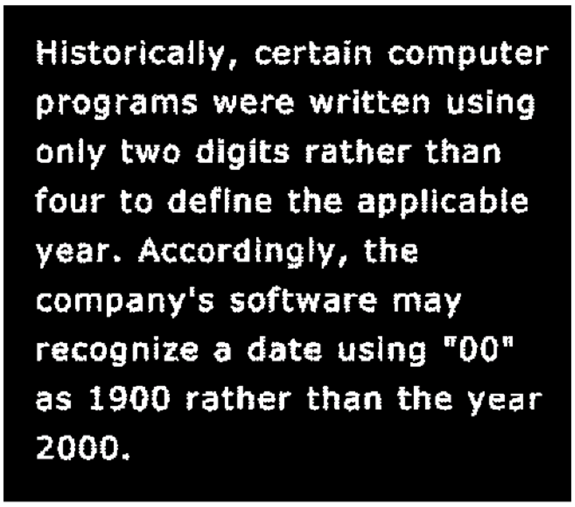
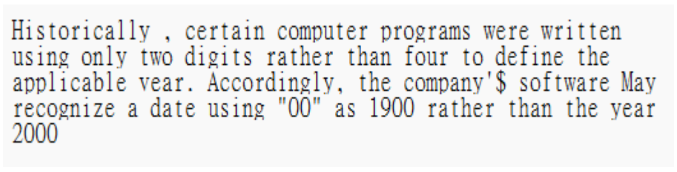
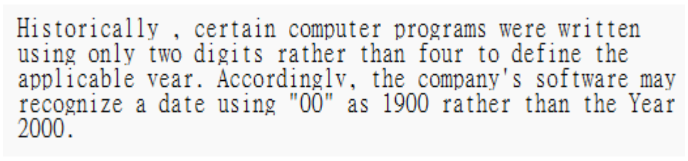

Historically, certain computer programs were written using only two digits rather than four to define the applicable year. Accordingly, the company's software may recognize a date using "00" as 1900 rather than the year 2000.

Fig 1-b Restored Picture



Historically , certain computer programs were written
using only two digits rather than four to define the
applicable vear. Accordingly, the company'$ software May
recognize a date using "00" as 1900 rather than the year
2000



Historically , certain computer programs were written
using only two digits rather than four to define the
applicable vear. Accordinglv, the company's software may
recognize a date using "00" as 1900 rather than the Year
2000.

Fig 1-c Original picture recognition

Fig 1-d Restored picture recognition

We observed a slight difference after applying Morphological Dilation (MD). For instance, the original recognition **misidentified** “s” as “$”, **whereas** the post-MD process **correctly recognized** it as “s”.

Code:

```
import cv2
import numpy as np
import easyocr
import os

INPUT_FILE = "text-broken.tif"

# 輸出檔名
OUT_IMG_FIXED = "broken_restored.tif"        # 修復後的圖片
OUT_TXT_ORIG   = "text broken_original.txt" # 原圖的 OCR 結果 (預期會很糟)
OUT_TXT_FIXED = "text broken_fixed.txt"      # 修復後的 OCR 結果 (預期會變好)

CONFIG = {
    # 修復參數
    "threshold_value": 127,
    "kernel_size": 3,
    "iterations": 1,
    "mode": "dilate", # 'dilate' or 'close'

    "invert_saved_image": False
}

def run_comparison():
    reader = easyocr.Reader(['en'], gpu=False) # 有顯卡可改 gpu=True
    img_original = cv2.imread(INPUT_FILE, 0)
    # Part A: Original
    try:
        results_orig = reader.readtext(img_original, detail=0, paragraph=True)
        text_orig = "\n".join(results_orig)

        with open(OUT_TXT_ORIG, 'w', encoding='utf-8') as f:
            f.write(text_orig)
        if not text_orig.strip():
            print(" serious broken")
    except Exception as e:
        print(f"原圖辨識發生錯誤: {e}")
    # ==========================================
    # Part B: Exp
    # ==========================================
    _, binary = cv2.threshold(img_original, CONFIG["threshold_value"], 255, cv2.THRESH_BINARY)
    k_size = CONFIG["kernel_size"]
    kernel = np.ones((k_size, k_size), np.uint8)

    if CONFIG["mode"] == "dilate":
        processed = cv2.dilate(binary, kernel, iterations=CONFIG["iterations"])
    else:
        processed = cv2.morphologyEx(binary, cv2.MORPH_CLOSE, kernel,
iterations=CONFIG["iterations"])

    # B3. 存圖片 (給報告用)
    if CONFIG["invert_saved_image"]:
        cv2.imwrite(OUT_IMG_FIXED, cv2.bitwise_not(processed))
    else:
        cv2.imwrite(OUT_IMG_FIXED, processed)
    print(f"-> 修復後的圖片已儲存: {OUT_IMG_FIXED}")

    # B4. 辨識修復後的圖片
    results_fixed = reader.readtext(processed, detail=0, paragraph=True)
    text_fixed = "\n".join(results_fixed)

   # B5. 存文字
    with open(OUT_TXT_FIXED, 'w', encoding='utf-8') as f:
        f.write(text_fixed)
    print(f"-> 修復後辨識結果已儲存: {OUT_TXT_FIXED}")

    print("\n" + "="*40)
    print("     *** 差異比對 (前 100 字預覽) ***")
    print("="*40)
    print(f"[原圖 Original]:\n{text_orig[:100]}...")
    print("-" * 40)
    print(f"[修復 Fixed]:\n{text_fixed[:100]}...")
    print("="*40)

if __name__ == "__main__":
    run_comparison()
```

2. Please recognize the image: text.tif. Please describe your method, procedures, final text file and print out the source code?

There are four function blocks in Python code. Here are the settings:

Block name	parameter
Binary Thresholding (BT)	Threshold = 127, Type = THRESH_BINARY
Connected Components(CCL)	Method = cv2.connectedComponents
Border Clearing Algo(BCA)	Target = Objects touching boundary
EasyOCR(OCR)	Language = 'en', Mode = paragraph

Path of Each Picture:

Path 1: Original Image Recognition (Control Group)

Input Image -> OCR -> Text Output

Path 2: Text Restoration & Recognition (Experimental Group)

Input Image -> BT -> CCL -> BCA -> Saved Image -> OCR -> Text Output

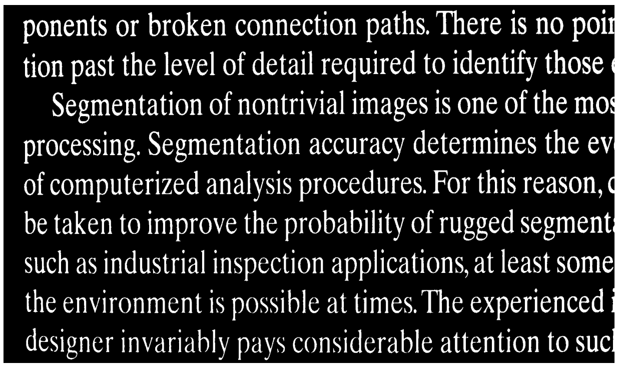

ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mos
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason, c
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced i
designer invariably pays considerable attention to suc

Fig 2-a Original Picture

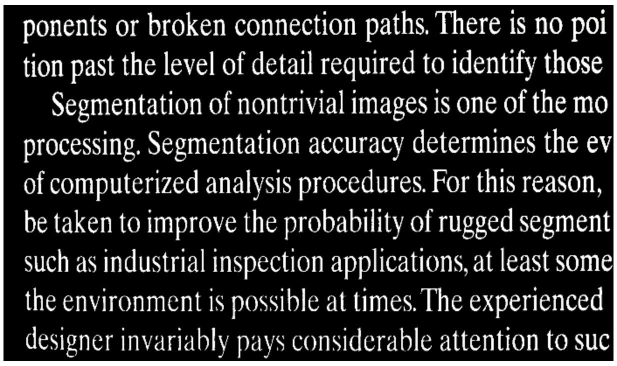
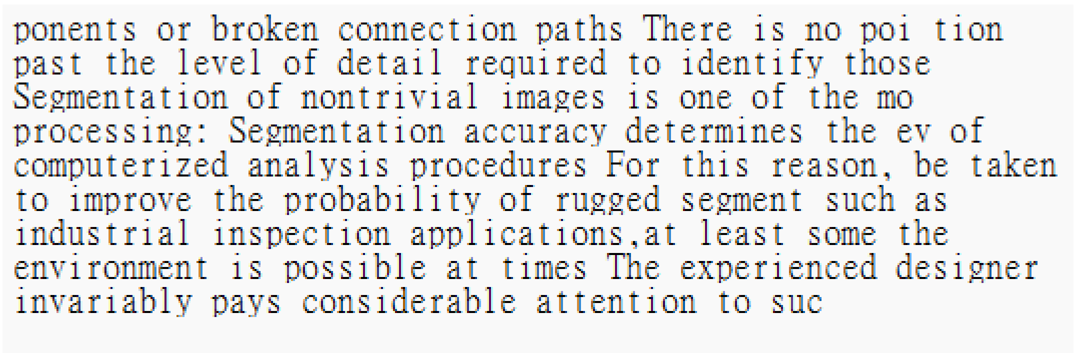

ponents or broken connection paths. There is no poi
tion past the level of detail required to identify those
Segmentation of nontrivial images is one of the mo
processing. Segmentation accuracy determines the ev
of computerized analysis procedures. For this reason,
be taken to improve the probability of rugged segment
such as industrial inspection applications, at least some
the environment is possible at times. The experienced
designer invariably pays considerable attention to suc

Fig 2-b Picture after deleting words which touch boundary



```
ponents or broken connection paths There is no poi tion
past the level of detail required to identify those
Segmentation of nontrivial images is one of the mo
processing: Segmentation accuracy determines the ev of
computerized analysis procedures For this reason, be taken
to improve the probability of rugged segment such as
industrial inspection applications,at least some the
environment is possible at times The experienced designer
invariably pays considerable attention to suc
```

Fig 2-c Recognition after deleting words which touch boundary

We observed that text elements touching the image **boundary** were **removed**. For instance, the string “poin” was **truncated** to “poi” as the character touching the edge was deleted.

Code:

```python
import cv2
import numpy as np
import easyocr
import os

# ==========================================
# 參數設定
# ==========================================
INPUT_FILE = "text.tif"
OUTPUT_FILE = "text_cleared_manual.tif"
OUTPUT_TXT    = "text.txt"

CONFIG = {
    # 二值化參數 (黑底白字模式)
    "threshold_value": 127,

    # 存檔風格: False=黑底白字(原汁原味), True=白底黑字
    "invert_saved_image": False
}

def my_clear_border(binary_img):
    """
    手刻的邊緣清除演算法
    輸入: 二值化影像 (文字=255, 背景=0)
    輸出: 清除邊緣文字後的二值化影像
    """
    # 1. 連通組件標記 (Find Connected Components)
    # num_labels: 總共有幾個物件 (包含背景 0)
    # labels: 一張跟原圖一樣大的圖，每個像素的值是它所屬的物件 ID
    num_labels, labels = cv2.connectedComponents(binary_img)

    print(f"   -> 偵測到 {num_labels - 1} 個連通物件")

    # 取得影像尺寸
    h, w = binary_img.shape

    # 2. 建立一個集合來記錄「要被刪除的 ID」
    ids_to_remove = set()
    # 3. 掃描影像邊界，找出碰到邊界的物件 ID
    # 這裡我們只看四個邊框的像素，不用掃描整張圖，效率比較高

    # 上邊界 (Row 0)
    top_row_ids = np.unique(labels[0, :])
    ids_to_remove.update(top_row_ids)

    # 下邊界 (Row h-1)
    bottom_row_ids = np.unique(labels[h-1, :])
    ids_to_remove.update(bottom_row_ids)

    # 左邊界 (Col 0)
    left_col_ids = np.unique(labels[:, 0])
    ids_to_remove.update(left_col_ids)

    # 右邊界 (Col w-1)
    right_col_ids = np.unique(labels[:, w-1])
    ids_to_remove.update(right_col_ids)

    # 移除 ID 0 (背景)，因為背景碰到邊框是正常的，不能刪掉背景
    if 0 in ids_to_remove:
        ids_to_remove.remove(0)

    print(f"   -> 共有 {len(ids_to_remove)} 個物件接觸邊界，將被移除。")

    # 4. 重建影像 (Reconstruct)
    # 建立一個遮罩：如果像素的 ID 在「刪除名單」中，設為 True
    # np.isin 可以快速檢查 labels 中的每個值是否在 ids_to_remove 裡
    mask_to_remove = np.isin(labels, list(ids_to_remove))

    # 複製一份原圖
    output_img = binary_img.copy()

    # 將遮罩對應的位置設為 0 (黑色)
    output_img[mask_to_remove] = 0

    return output_img

def process_manual_clear():
    # 1. 準備 OCR
    print("--- 正在載入 EasyOCR... ---")
    reader = easyocr.Reader(['en'], gpu=False)

    # 2. 讀取影像
    if not os.path.exists(INPUT_FILE):
        print(f"錯誤: 找不到檔案 {INPUT_FILE}")
        return
    img = cv2.imread(INPUT_FILE, 0)

    # 3. 二值化 (文字=白, 背景=黑)
    _, binary = cv2.threshold(img, CONFIG["threshold_value"], 255, cv2.THRESH_BINARY)

    # 4. 執行我們自己寫的演算法
    print("--- 執行自製邊緣清除演算法 (My Clear Border) ---")
    cleaned_img = my_clear_border(binary)
    # 5. 存檔 (圖片)
    if CONFIG["invert_saved_image"]:
        cv2.imwrite(OUTPUT_FILE, cv2.bitwise_not(cleaned_img))
    else:
        cv2.imwrite(OUTPUT_FILE, cleaned_img)
    print(f"-> 圖片已儲存: {OUTPUT_FILE}")

    # 6. OCR 辨識
    print("--- 正在辨識最終結果... ---")
    results = reader.readtext(cleaned_img, detail=0, paragraph=True)
    text = "\n".join(results)

    with open(OUTPUT_TXT, 'w', encoding='utf-8') as f:
        f.write(text)
    print(f"-> 文字已儲存: {OUTPUT_TXT}")

    # 簡單驗證
    if len(text) > 0:
        print(f"辨識成功！(長度: {len(text)})")
    else:
        print("警告: 辨識結果為空，請檢查圖片是否變全黑。")

if __name__ == "__main__":
    process_manual_clear()
```

111511167 黃弘智

3. Please recognize the image: text-sineshade.tif. Please describe your method, procedures, final text file and print out the source code?

There are four function blocks in Python code. Here are the settings:

Block name	parameter
Zigzag Scan(ZS)	Mode = Bidirectional
Moving Average (MA)	Window Size (n) = 20
Thresholding (TH)	Scale Factor (c) = 0.5
PyLaia model (OCR)	CRNN = CNN + LSTM + CTC

Path of Each Picture:

Path 1: Original Image Recognition (Control Group)

Input Image -> OCR -> Text Output (Cannot find anything)

Path 2: Sine Shade Text Restoration & Recognition (Experimental Group)

Input Image -> ZS -> MA -> TH -> Reshape -> Clip -> OCR

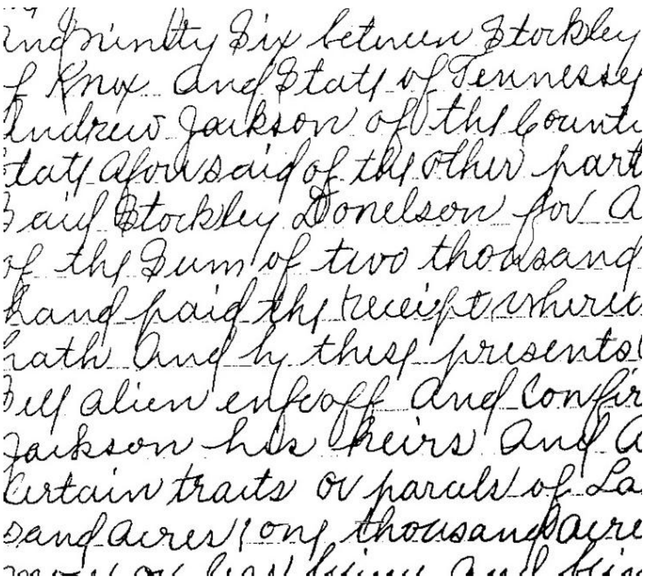

ind nintty Six between Stokley
f Knox and Stat of Tennessee
Andrew Jackson of the Counte
tate aforesaid of the other part
Said Stockley Donelson for A
of the Sum of two thousand
hand paid the receipt whereo
hath and by these presents
sell alien enfeoff and Confir
Jackson his heirs and a
certain traits or parcels of La
sand acres one thousand ayre

Fig 3-a Picture after removing sineshade

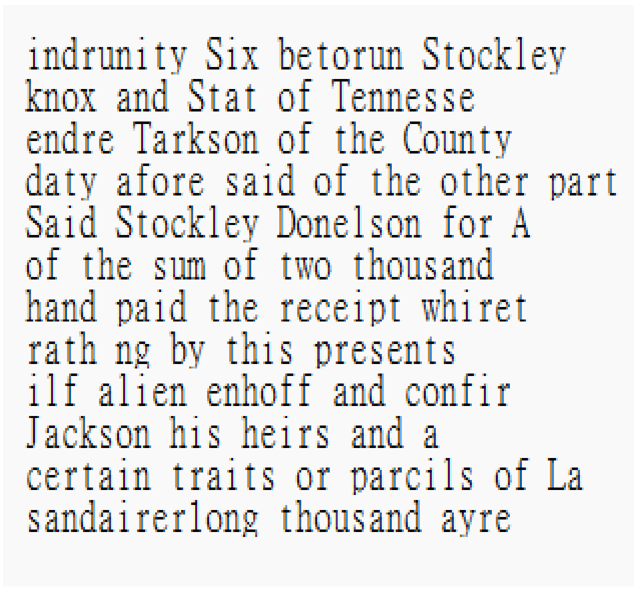

indrunity Six betorun Stockley
knox and Stat of Tennesse
endre Tarkson of the County
daty afore said of the other part
Said Stockley Donelson for A
of the sum of two thousand
hand paid the receipt whiret
rath ng by this presents
ilf alien enhoff and confir
Jackson his heirs and a
certain traits or parcils of La
sandairerlong thousand ayre

Fig 3-b OCR via PyLaia model

(in “transkribus”)

Code:

```
import cv2
import numpy as np
import os

# ==========================================
#  參數設定
# ==========================================
INPUT_FILE = "text-sineshade.tif"
OUTPUT_FILE = "sineshade_ma_restored.jpg"
OUTPUT_TXT = "text_sineshade_ma.txt"

#  使用者指定參數
# n:  計算平均的窗口大小  (通常設為筆畫寬度的  5  倍左右，這裡設  20)
# c:  閥值係數  (Threshold = Average * c)
N_MOVING_AVG = 20
C_FACTOR = 0.5

def moving_average_threshold(img, n, c):
    h, w = img.shape

    # 1.  將圖片轉為  Zigzag  序列  (避免每行開頭沒有歷史數據的斷層)
    #  偶數行:  左  ->  右
    #  奇數行:  右  ->  左
    img_flat = []
    indices_map = [] #  紀錄原始座標以便還原

    for r in range(h):
        row_data = img[r, :]
        row_indices = [(r, x) for x in range(w)]
        if r % 2 == 1:
            row_data = row_data[::-1]            #  反轉
            row_indices = row_indices[::-1] #  座標跟著反轉

        img_flat.extend(row_data)
        indices_map.extend(row_indices)

    img_flat = np.array(img_flat, dtype=float)

    # 2.  計算移動平均  (Moving Average)
    # m_k = (z_k + z_k-1 + ... + z_k-n+1) / n
    #  為了效率使用卷積  (Convolution)  計算
    window = np.ones(n) / n
    # mode='same'  會造成中心偏移，這裡我們用  'full'  取前段模擬歷史平均
    m_values = np.convolve(img_flat, window, mode='full')[:len(img_flat)]

    # 3.  進行二值化判斷
    #  閥值  T =  平均值  * c
    #  若  像素  < T (比背景暗很多) ->  文字  (0)
    #  若  像素  >= T ->  背景  (255)
    thresholds = m_values * c
    binary_flat = np.where(img_flat < thresholds, 0, 255).astype(np.uint8)

    # 4.  還原回  2D  圖片
    output_img = np.zeros((h, w), dtype=np.uint8)
   for i, (r, x) in enumerate(indices_map):
        output_img[r, x] = binary_flat[i]

    return output_img
```

```
def run_ma_process():
    # 1. 讀取
    if not os.path.exists(INPUT_FILE):
        print(f"錯誤: 找不到 {INPUT_FILE}")
        return
    img = cv2.imread(INPUT_FILE, 0)

    print(f"--- 執行 Moving Average Thresholding (n={N_MOVING_AVG}, c={C_FACTOR}) ---")

    # 2. 處理
    result = moving_average_threshold(img, N_MOVING_AVG, C_FACTOR)

    # 3. 存圖
    cv2.imwrite(OUTPUT_FILE, result)
    print(f"-> 圖片已儲存: {OUTPUT_FILE}")

if __name__ == "__main__":
    run_ma_process()
```

(OCR 則是在 https://www.transkribus.org/ai-text-recognition 執行)

4. Please recognize the image: text-spotshade.tif. Please describe your method, procedures, final text file and print out the source code?

There are four function blocks in Python code. Here are the settings:

Block name	parameter
Zigzag Scan(ZS)	Mode = Bidirectional
Moving Average (MA)	Window Size (n) = 20
Thresholding (TH)	Scale Factor (c) = 0.5
PyLaia model (OCR)	CRNN = CNN + LSTM + CTC

Path of Each Picture:

Path 1: Original Image Recognition (Control Group)

Input Image -> OCR -> Text Output (Cannot find anything)

Path 2: Spot shade Text Restoration & Recognition (Experimental Group)

Input Image -> ZS -> MA -> TH -> Reshape -> Clip -> OCR

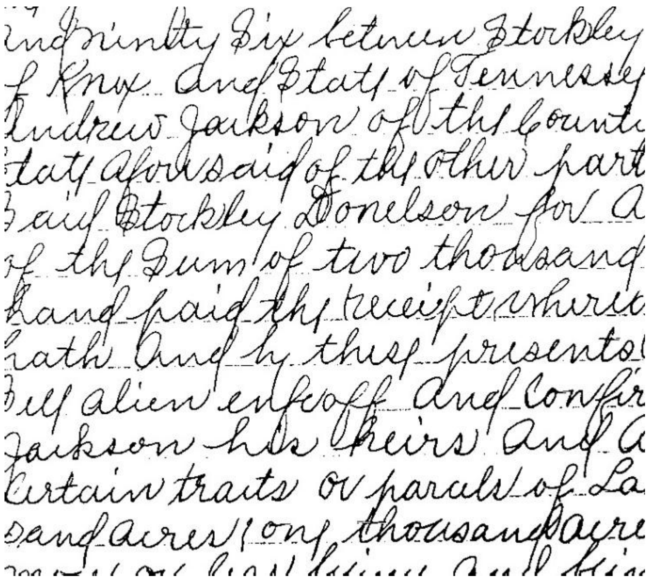

and ninety Six between Stockley
f Knox and State of Tennessee
Andrew Jackson of the Count
tate aforesaid of the other part
Said Stockley Donelson for A
f the Sum of two thousand
hand paid the receipt whereo
rath And by these presents
ell alien enfeoff and Confir
Jackson his heirs and a
certain traits or parcels of La
sand acres (one thousand acre

Fig 4-a Picture after removing spotshade

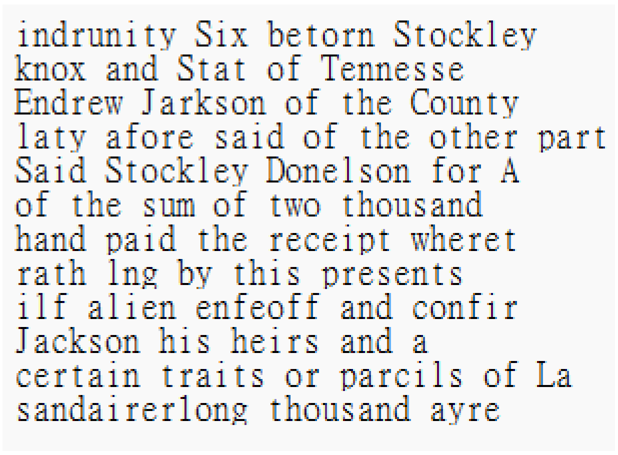

```
indrunity Six betorn Stockley
knox and Stat of Tennesse
Endrew Jarkson of the County
laty afore said of the other part
Said Stockley Donelson for A
of the sum of two thousand
hand paid the receipt wheret
rath lng by this presents
ilf alien enfeoff and confir
Jackson his heirs and a
certain traits or parcils of La
sandairerlong thousand ayre
```

Fig 4-b OCR via PyLaia model

(in "transkribus")

111511167　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　　黃弘智

Code:

```
import cv2
import numpy as np
import easyocr
import os

INPUT_FILE = "text-spotshade.tif"
OUTPUT_FILE = "spotshade_restored.jpg"
OUTPUT_TXT = "text_spotshade.txt"
N_MOVING_AVG = 20      # n = 20
C_FACTOR = 0.5          # c = 0.5 (T = 50% of local average)

def moving_average_threshold(img, n, c):
    h, w = img.shape

    img_flat = []
    indices_map = []

    for r in range(h):
        row_data = img[r, :]
        row_indices = [(r, x) for x in range(w)]
        if r % 2 == 1:
            row_data = row_data[::-1]
            row_indices = row_indices[::-1]

        img_flat.extend(row_data)
        indices_map.extend(row_indices)

    img_flat = np.array(img_flat, dtype=float)

    # 2. 計算移動平均 (Moving Average)
    # 利用卷積快速計算過去 n 點的平均
    window = np.ones(n) / n
    # mode='full' 取前段，模擬歷史平均
    m_values = np.convolve(img_flat, window, mode='full')[:len(img_flat)]

    # 3. Dynamic Thresholding
    thresholds = m_values * c
    binary_flat = np.where(img_flat < thresholds, 0, 255).astype(np.uint8)

    # 4. Reshape
    output_img = np.zeros((h, w), dtype=np.uint8)
    for i, (r, x) in enumerate(indices_map):
        output_img[r, x] = binary_flat[i]

    return output_img

def run_spotshade_process():
    img = cv2.imread(INPUT_FILE, 0)
    print(f"--- 正在處理 {INPUT_FILE} (n={N_MOVING_AVG}, c={C_FACTOR}) ---")
    result = moving_average_threshold(img, N_MOVING_AVG, C_FACTOR)
    cv2.imwrite(OUTPUT_FILE, result)

if __name__ == "__main__":
    run_spotshade_process()
```

(OCR 則是在 https://www.transkribus.org/ai-text-recognition 執行)