數位影像處理
期末應用實作報告

題目: 多功能影像編輯工具

姓名:陳宜謙

學號:409410064

日期:2022/06/21

目錄

製作動機

起初是想要做一個能夠將白色背景去除變成透明的工具，因為在製作簡報時，有時在網路上下載的素材，是有包含白色底圖的，就常常還要透過網路線上的去背工具做使用，但因為網路上這些的工具，常常夾帶很多廣告或有張數上的限制，下載照片後也不知道會不會夾帶了病毒一起進到電腦中，所以才自己製作了這套小工具來使用，開發過程中也將其他實用的功能也加進來，讓這套工具更完善。

功能介紹

總共有 5 種功能，依序介紹

雙邊濾波

能夠將照片中的雜訊去除，並且保留眼睛、鼻子等邊緣資訊，可以用來實現美膚的效果，若將濾波大小和 sigma 值增大，會有模糊化的效果。

去背

本次主要實做的功能，能夠將白色背景去除變成透明，方便之後用來製作簡報、卡片等其他用途。

調整曝光

若在拍照時，因為當下環境的因素，導致照片過度曝光或曝光不足，可以使用此功能來改善影像的亮度。

影像對比

加強影像的對比能夠將影像中暗的地方變暗，亮的地方變亮，因為人類視覺系統對於影像對比的敏感度比絕對亮度高，因此通常覺得高對比的影像品質較佳。

影像銳化

我們往往在拍照時會受限於解析度的關係，導致拍出來的照片有些較為模糊的邊緣，所以利用銳化功能，將模糊的邊緣變得更銳利。

工作原理

主程式

建立圖形化介面的選單，讓使用者方便操作，藉由即時顯示調整後的影像，省去重複執行程式來調整參數帶來的不便，另外使用了 tkinter 的 filedialog 函式來存取要讀入或存檔的影像，不需要在程式中修改影像位置了。

雙邊濾波

利用 cv2 所提供的 bilateralFilter 函式來實做，去抓取使用者所調整的兩個參數大小，來套用到影像上，可以依自己想要的效果調整區域大小的直徑和

sigma 值，若數值越大則效果越強烈。

去背

先將照片從 BGR 轉成 BGRA，變成 png 的格式，多出來的 A 為透明度數值，另外再產生一張灰階的圖片，原本圖片中白色的部分轉成灰階後會落在靠近 255 的地方，使用者可以調整想要去背的門檻值，在門檻值以上的地方就會做透明化處理，使用 255 - 灰階數值 可以將一些邊緣的像素變成半透明，避免太過鋸齒的邊緣，最後呈現在畫面上的圖片，去背部分以黃色來顯示。

調整曝光

利用之前上課所學的 gamma_correction 的轉換函數，透過調整不同的 gamma 數值來改變影像的亮度，gamma 值在等於 1 時，影像強度不變，大於 1 時會使影像變暗，反之，小於 1 時會使影像變亮。

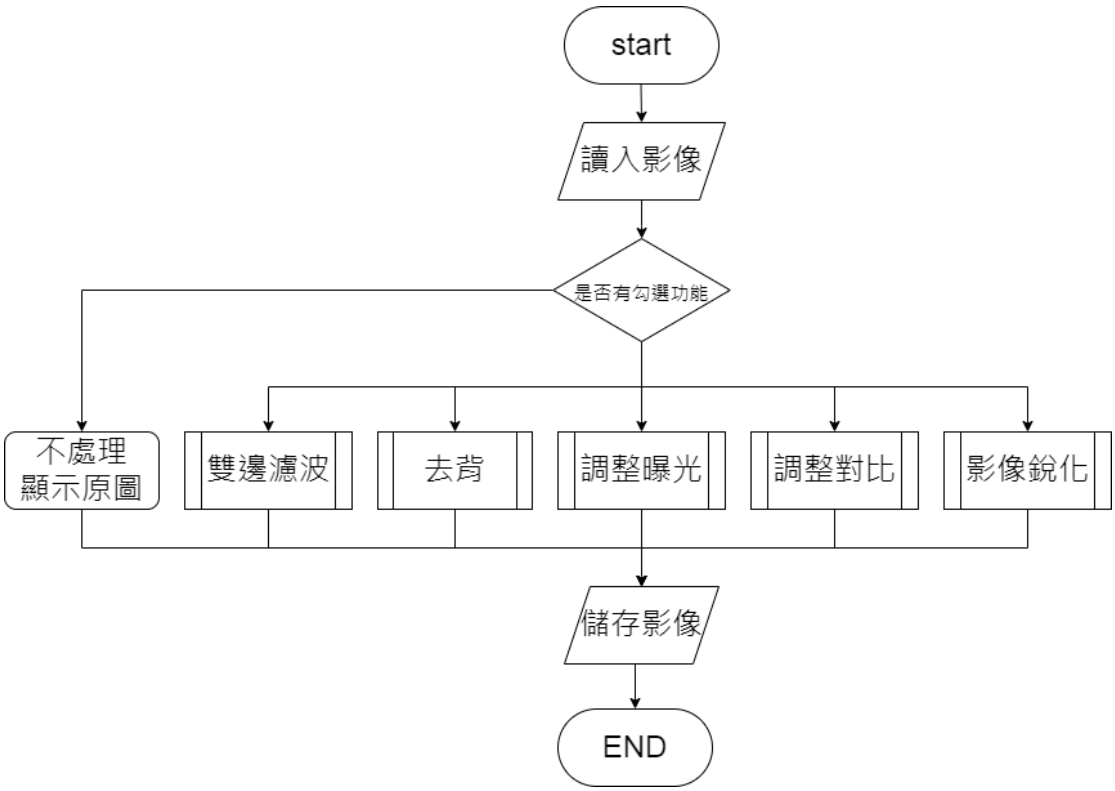
影像對比

使用 beta_correction 的轉換函數，調整 a,b 的數值來改變影像的對比，當 a,b 大於 1 時會增強影像對比，若 a,b 小於 1 時會減弱影像對比。

影像銳化

銳化的功能實現，其實就是模糊的反操作，首先使用 cv2 的 GaussianBlur 產生高斯模糊圖片，之後再使用 cv2 的 addWeighted，以 原圖:模糊圖片 =1.5 : -0.5 的比例進行混合，就能得到銳化後的圖片了。

程式流程



程式碼

完整程式碼: https://github.com/YiCian-Chen/DIP

主程式

主程式的物件這邊以雙邊濾波做為範例,其他建立物件是相同的原理就不重複附上

```
# 建立視窗
root = tk.Tk()
root.geometry('350x500')
root.title('圖片處理')

# 建立載入/存檔按鈕
tk.Button(root, text='選擇圖片', command=open_file).grid(
      column=0, row=0, padx=10, pady=10)
tk.Button(root, text="儲存圖片", command=save_file).grid(
      column=1, row=0, padx=10, pady=10)

################ Bilateral 雙邊濾波
# Button 物件是否勾選時的變數
Bilateral_var = tk.IntVar()
# 建立 Button 物件
Bilateral_btn = tk.Checkbutton(root, text='雙邊濾波', command=lambda:
check(Bilateral_btn),var=Bilateral_var)
# 設定 Button 物件的位置
Bilateral_btn.grid(column=0, row=1, padx=10, pady=10)
# 建立 Scale 物件
Bilateral_size_scale = tk.Scale(orient='horizontal', label='size',
    command=lambda x: Bilateral_func(Bilateral_size_scale, Bilateral_sigma_scale))
# 設定 Scale 物件數值大小的界線
Bilateral_size_scale.config(from_=1, to=30, tickinterval=29)
# 初始化 Scale 物件的數值
Bilateral_size_scale.set(1)
# 設定 Scale 物件的位置
Bilateral_size_scale.grid(column=1, row=1)

Bilateral_sigma_scale = tk.Scale(orient='horizontal', label='sigma',
    command=lambda x: Bilateral_func(Bilateral_size_scale, Bilateral_sigma_scale))
```

```
Bilateral_sigma_scale.config(from_=1, to=200, tickinterval=199)
Bilateral_sigma_scale.set(1)
Bilateral_sigma_scale.grid(column=2, row=1)
# 查看是哪一項功能被選取
def check(item):
    if item['text'] == "雙邊濾波":
        # 將選取方塊勾選時
        if Bilateral_var.get() == 1:
            reset() # 將全部功能取消勾選
            Bilateral_var.set(1)  # 將此功能勾選
            # 讓滑桿物件可以使用
            Bilateral_sigma_scale['state'] = tk.NORMAL
            Bilateral_size_scale['state'] = tk.NORMAL
            # 呼叫函式將影像進行處理
            Bilateral_func(Bilateral_size_scale, Bilateral_sigma_scale)
        else:  # 選取方塊取消勾選時
            # 將滑桿物件停用
            Bilateral_sigma_scale['state'] = tk.DISABLED
            Bilateral_size_scale['state'] = tk.DISABLED
            default()  # 呼叫函式將影像恢復成原始影像

# 將全部的物件設定為停用狀態
def reset():
    Bilateral_sigma_scale['state'] = tk.DISABLED
    Bilateral_size_scale['state'] = tk.DISABLED
    Bilateral_var.set(0)

reset()
root.mainloop()
```

載入/存檔

```
# 儲存影像
def save_file():
    global tmp
    # 詢問影像儲存位置
    path = asksaveasfile(initialfile='output.png',
                         defaultextension=".png",
                         filetypes=[
                             ("All Files", "*.*"),
```

```
                            ("png files", "*.png"),
                            ("jpeg files", "*.jpg")])
    # 將影像寫入指定的位置
    cv2.imwrite(path.name, tmp)


# 載入影像
def open_file():
    global img,tmp
    # 詢問影像開啟位置
    path = askopenfilename(title='選擇',
                           filetypes=[
                               ('All Files', '*'),
                               ("jpeg files", "*.jpg"),
                               ("png files", "*.png")])
    # 將載入影像從指定的位置，透過解碼來解決讀入中文檔名問題
    img = cv2.imdecode(np.fromfile(path, dtype=np.uint8), -1)
    tmp = img
    cv2.imshow("picture",tmp)
```

雙邊濾波

```
# 雙邊濾波
def Bilateral_func(item1,item2):
    global img,tmp
    size = int(item1.get())
    sigma = int(item2.get())
    tmp = cv2.bilateralFilter(img,size,sigma,sigma)
    cv2.imshow('picture', tmp)
```

去背

```
# 去背函式
def background_func(val):
    global img,tmp
    val = int(val) # 將數值從 str 轉成 int
    tmp = cv2.cvtColor(img, cv2.COLOR_BGR2BGRA) # 因為是 jpg，要轉換顏色為 BGRA
    gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY) # 轉成灰階方便做出判斷
    tmp_show = tmp.copy()
    h, w = img.shape[:2]
    for x in range(w):
        for y in range(h):
            if gray[y, x] > val: # 若該像素大於門檻值
```

```
                tmp[y, x, 3] = 255 - gray[y, x] # 調整該像素位置的透明度
                tmp_show[y, x] = [0, 255, 255, 255] # 將有調整的地方用黃色顯示
                # 使用 255 - gray[y, x] 可以將一些邊緣的像素變成半透明，避免太過鋸齒的邊緣
    cv2.imshow('picture', tmp_show)
```

調整曝光

```
# 調整曝光
def gamma_func(val):
    global img, tmp
    val = float(val)
    tmp = gamma_correction(img, val)
    cv2.imshow('picture', tmp)


# 伽瑪矯正函式
def gamma_correction(f, gamma):
    g = f.copy()
    nr, nc = f.shape[:2]
    c = 255.0 / (255.0 ** gamma)
    table = np.zeros(256)
    for i in range(256):
        table[i] = round(i ** gamma * c, 0)
    if f.ndim != 3:
        for x in range(nr):
            for y in range(nc):
                g[x, y] = table[f[x, y]]
    else:
        for x in range(nr):
            for y in range(nc):
                for k in range(3):
                    g[x, y, k] = table[f[x, y, k]]
    return g
```

影像對比

```
# 調整對比
def beta_func(item1, item2):
    global img, tmp
    A = float(item1.get())
    B = float(item2.get())
    tmp = beta_correction(img,A,B)
    cv2.imshow('picture', tmp)
```

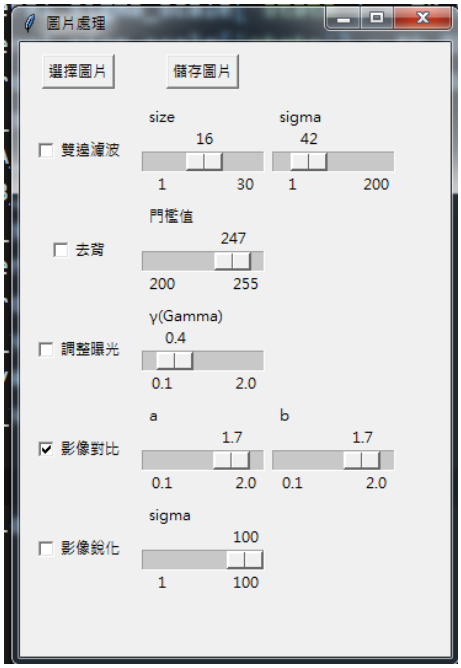
```
# Beta 矯正函式
def beta_correction(f, a, b):
    g = f.copy()
    nr, nc = f.shape[:2]
    x = np.linspace(0, 1, 256) # 在 0~1 之間產生 256 個點
    # 將不完整 Beta 函數輸出正規化至 0~255
    table = np.round(special.betainc(a, b, x) * 255, 0)
    if f.ndim != 3:
        for x in range(nr):
            for y in range(nc):
                g[x, y] = table[f[x, y]]
    else:
        for x in range(nr):
            for y in range(nc):
                for k in range(3):
                    g[x, y, k] = table[f[x, y, k]]
    return g
```

影像銳化

```
# 影像銳化
def sharp_func(val):
    global img, tmp
    val = int(val)
    blur = cv2.GaussianBlur(img, (0, 0), val) # 先將影像模糊
    tmp = cv2.addWeighted(img, 1.5, blur, -0.5, 0) # 模糊的反操作來實現銳化
    cv2.imshow('picture', tmp)
```

執行結果

操作介面

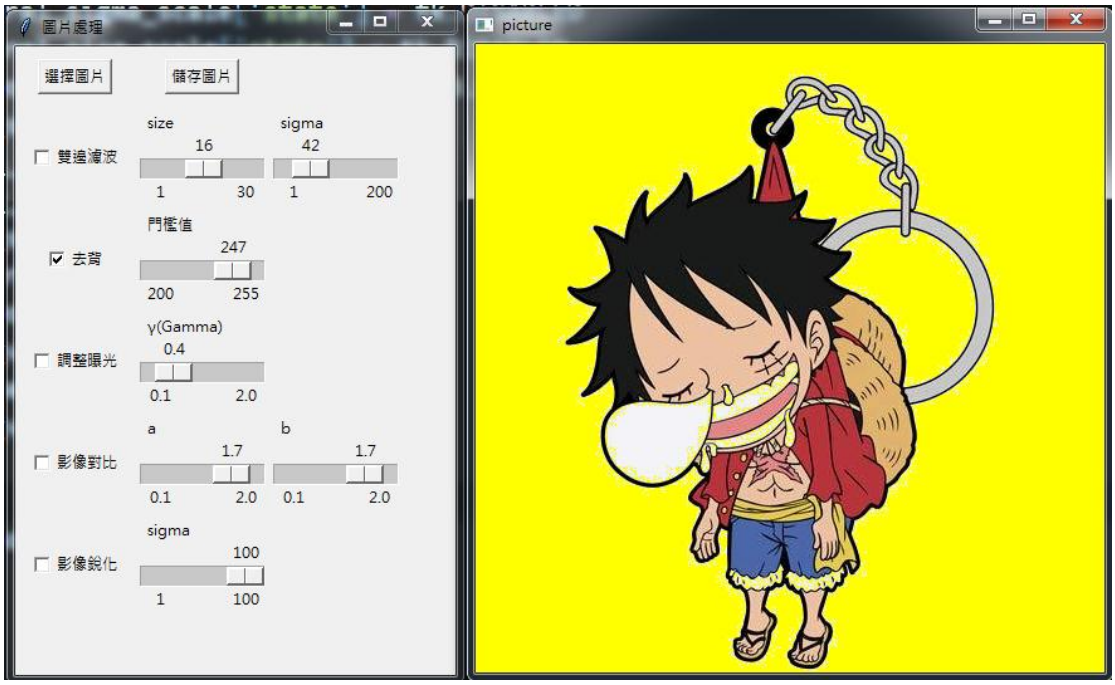


雙邊濾波

進行了平滑化處理，但保留了眼睛鼻子等特徵

去背

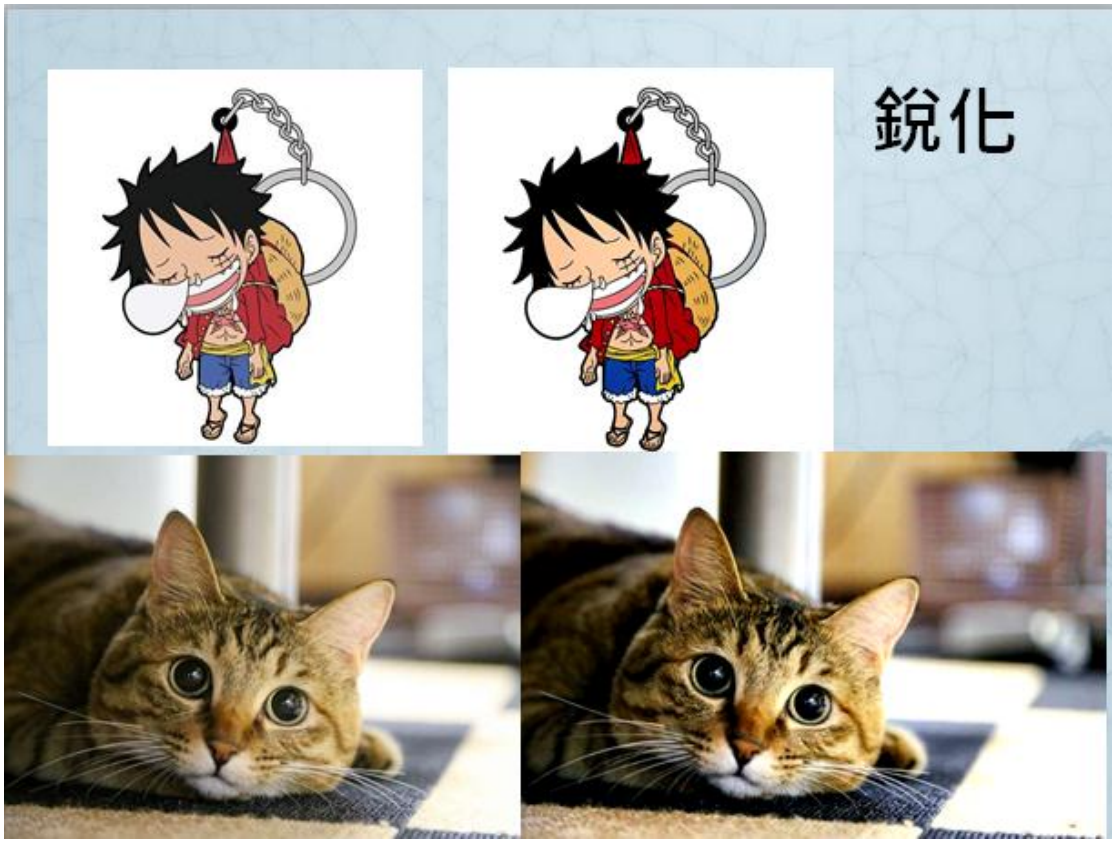
預覽圖片去除背景的部分呈現黃色



調整曝光

影像對比

影像銳化



問題討論與心得

在實作這次的題目時，我覺得最複雜的地方就是 GUI 界面的條件判斷那部分，因為想要製作一個簡單且方便使用的應用程式，所以也將讀入影像的部分，從以前要自己從程式碼當中改掉路徑檔名，修改成只要按個按鈕，就能用圖形化界面的方式，找到自己存放在電腦的影像，儲存時也能再依據自己的喜好和需求，隨時更改存檔的路徑和檔名。

製作過程中也要加入許多防呆的條件判斷，盡可能的將各項功能都做到很簡潔明瞭，讓不懂程式或原理的人也能透過這套應用程式來進行影像編輯。

因為顯示的圖像為即時呈現的，所以如果照片選用的太大張，執行各項功能時，所需要的時間也會越久，原本有想要再加入能夠改變色調的功能，但因為在測試時小張的圖片，每一筆計算就會花上不少時間，在使用上會有卡頓的感覺，影響到使用體驗，所以最後才沒有這項功能。如果要解決卡頓的問題，第一個解決方法可以先進行演算法的優化，避免一直重複迴圈，降低時間複雜度，可以加快計算的時間，第二個解決方法就是不要使用即時顯示的方式，可以改用按鈕點擊才顯示的方式，可以將低計算的次數。

在去背功能的部分，原本是將門檻值寫死固定的，但是發現如果有像範例的魯夫圖片那樣，在牙齒或其他部分有偏白色出現，就需要進行調整門檻值，讓這些偏白色的部份能夠保留住。

成果影片

https://youtu.be/w9kGpPE_hNo

參考資料

GUI 模組 tkinter
http://yhhuang1966.blogspot.com/2021/09/python-gui-tkinter-scale.html

影像銳化
https://blog.csdn.net/elegentbeauty/article/details/79849887

數位影像處理課本
https://www.books.com.tw/products/0010866466?sloc=main