Ingeniería en computación

Procesamiento de imágenes

Profesor: Edith Cristina Herrera Luna

Alumnos:
Cesar Augusto Ramírez Adán
Rodrigo Navarrete Moreno
Adrian Parada López

Documentación del proyecto

Implementación de procesamiento de imágenes a la montura y redimensión de tatuajes en una parte del cuerpo

Fecha: 02 de junio de 2022

Introducción

El procesamiento de imágenes en los últimos años se ha convertido en algo demasiado importante en los últimos años, ejemplo de ello: rayos x, mejoramiento de imágenes antiguas, montaje en paisajes. Cada técnica se ha ido mejorando durante el tiempo para esto se crearon mucho software en donde facilita el trabajo. Pero la pregunta es, ¿Quién desarrolla esos sistemas?
Existen muchos campos en donde se puede desarrollar ese tipo de software, materias a nivel licenciatura y algunas especialidades, en donde se comienza desde cero, explicando como funciona cada herramienta y como es posible generar nuevas aplicaciones, por ejemplo una aplicación que tendrá la funcionalidad de colocar tatuajes automáticamente en donde el cliente lo solicite, en este caso se llevan a cabo varios procesos que se vieron en la clase de procesamiento de imágenes, en donde nos otorgaron los fundamentos correspondientes para llevarla a cabo, todo el proceso realizado fue con base a lo aprendido en clase y también investigaciones por parte de cada integrante del equipo, aportando así nuevas ideas y logrando optimizar el código.
Todo el proceso fue desarrollado en el lenguaje de programación Python, ya que nos brindaba todas las herramientas necesarias que serian utilizadas en el desarrollo del software.
A continuación, se dará una breve explicación de como es que funciona el programa y de que pasos se llevaron a cabo para lograr el correcto funcionamiento del mismo.

Desarrollo

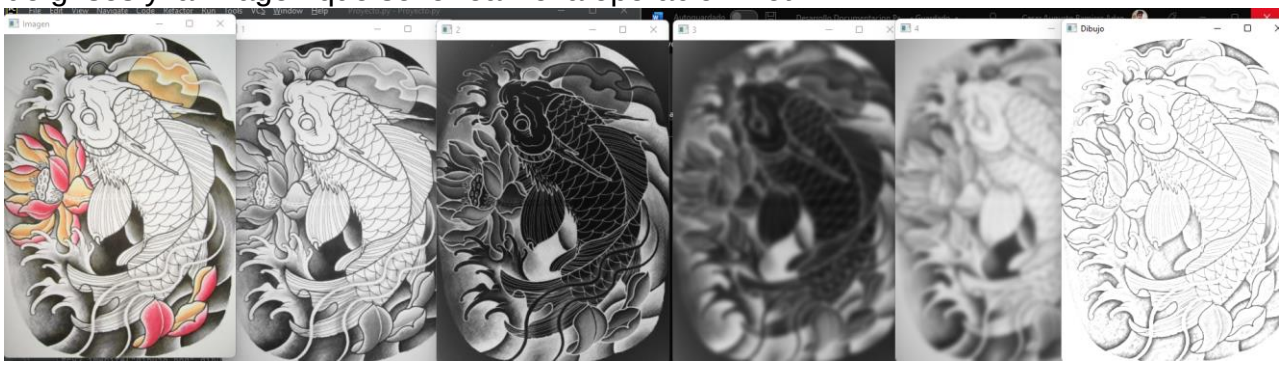
```
#Lectura de Imagenes
cpr=cv2.imread("11.jpg")#Parte del cuerpo
re = cv2.imread("d11.jpg") #Imagen a convertir dibujo
height, width = re.shape[:2]
```

En este espacio se realiza las lecturas de las imágenes (parte del cuerpo, y diseño de tatoo) además la información del tamaño del dibujo en 2 variables que usaremos más adelante.

```
#Conversion a dibujo
image= cv2.resize(re, None, fx=3/4, fy=3/4, interpolation=cv2.INTER_AREA)
grey = cv2.cvtColor(image,cv2.COLOR_BGR2GRAY)
invert = cv2.bitwise_not(grey)
blur=cv2.GaussianBlur(invert,(21,21),0)
invertedblur= cv2.bitwise_not(blur)
dibujo=cv2.divide(grey,invertedblur, scale=256.0)
cv2.imshow("Imagen",image)
cv2.imshow("Dibujo",dibujo)
#cv2.imwrite("dibujo.png",dibujo)
cv2.waitKey(0)
```

Esta parte del código es la que convertirá cualquier imagen que decida cargar el usuario a un dibujo en blanco y negro para el cual se aplicará en la simulación del como quedaría el tatuaje. Primero realizamos una redimensión a la imagen de ¾ de tamaño para un mejor manejo de las ventanas, después se transforma a escala de grises, se le invierten los colores, enseguida se le aplica un filtro gaussiano a la imagen ya con los colores invertidos y se guarda en nuestra variable “blur”.

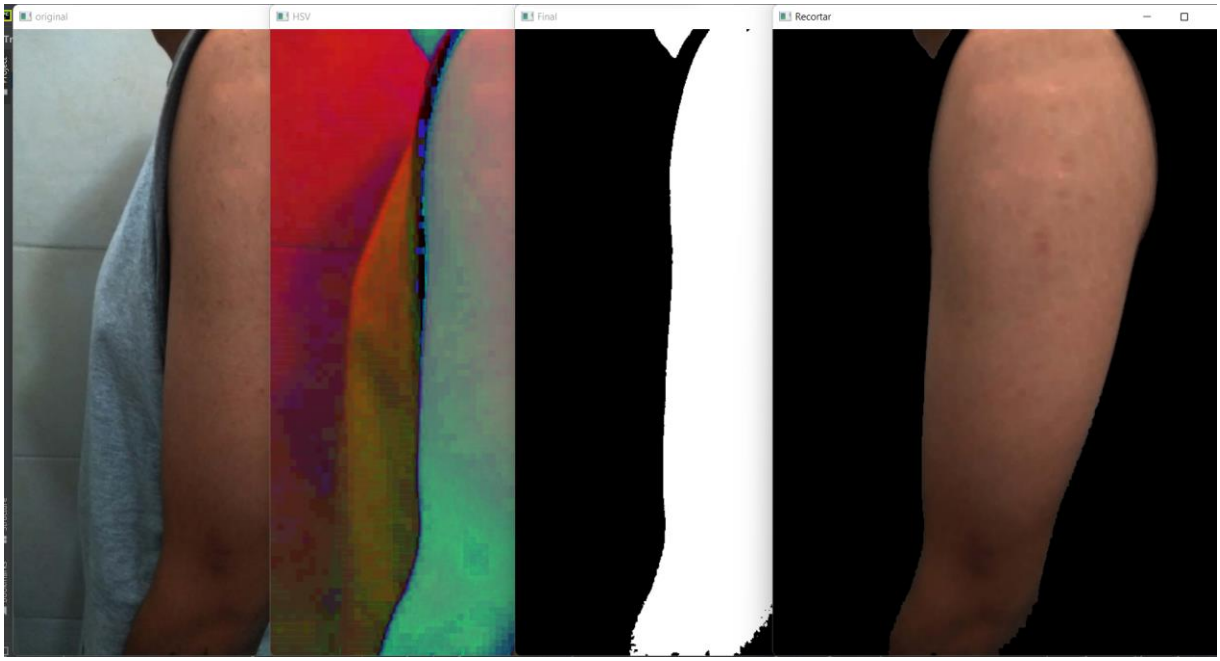
A nuestra variable “blur” aplicarle el operador Not para así cambiar su entrada de verdaderas a falsa y viceversa, para al final realizar una operación de división entre la imagen en escala de grises y la imagen que se le realizo la operación Not.



En el caso de la parte del cuerpo que el usuario cargue, se le realizo una segmentación para así poder obtener solo la parte de la piel donde se quiere realizar el tatuaje.

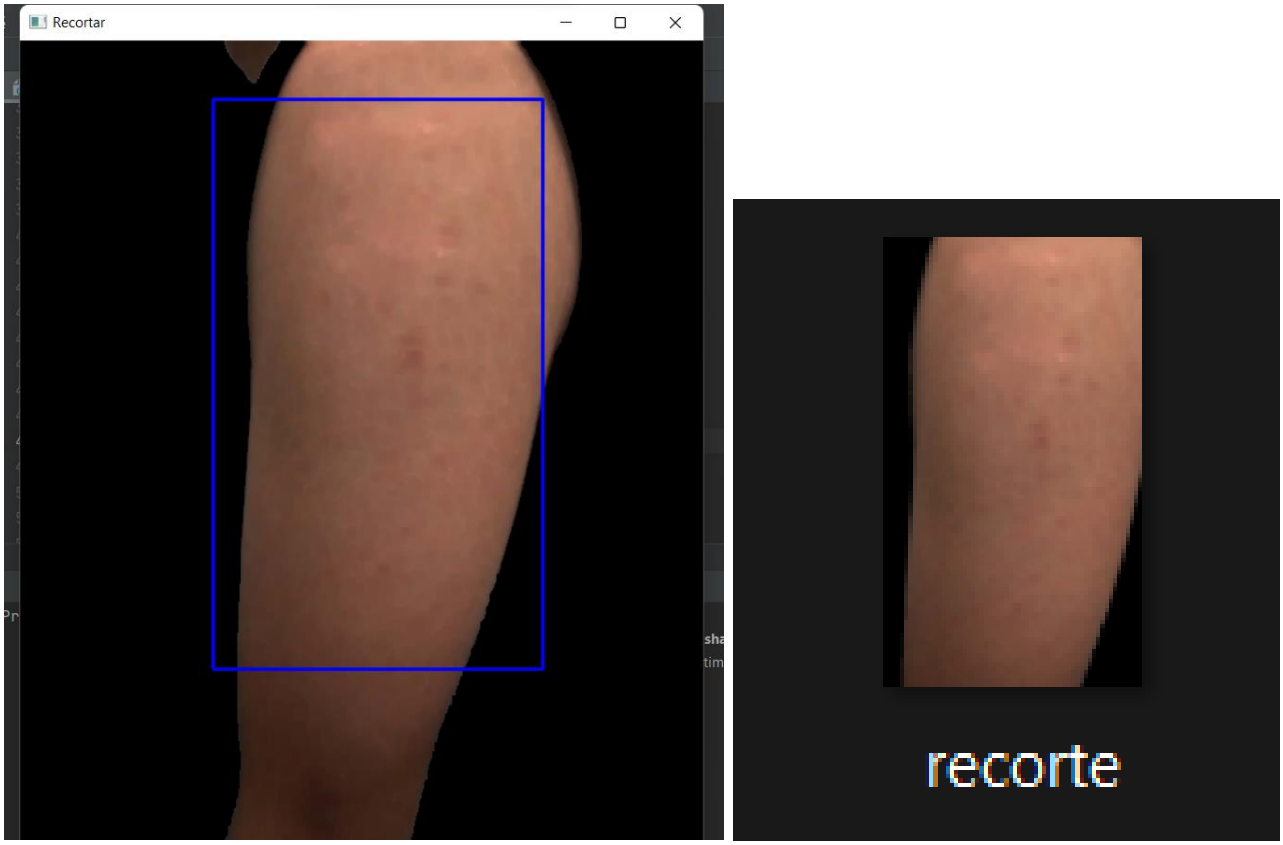
```
#segmentar
seg = cpr
seg2 = cv2.resize(seg, None, fx=1 / 2, fy=1 / 2, interpolation=cv2.INTER_AREA)
#Detectar color
hsv= cv2.cvtColor(seg2,cv2.COLOR_RGB2HSV)

lower_verde= np.array([245,255,221])
upper_verde= np.array([90,49,37])


mask=cv2.inRange(hsv,upper_verde, lower_verde)

piel=cv2.bitwise_and(seg2,seg2, mask = mask)
kernel = np.ones((6,6),np.uint8)
apertura=cv2.morphologyEx(piel,cv2.MORPH_CLOSE,kernel)
cv2.imshow("original",seg2)
cv2.waitKey(0)
cv2.imshow("HSV",hsv)
cv2.waitKey(0)
cv2.imshow("Final",mask)

cv2.waitKey(0)
#cv2.imshow("1",apertura)
cv2.imwrite("final.png",apertura)
cv2.waitKey(0)
```

Como en el anterior caso se le realiza un redimensión al imagen a la mitad de su tamaño para así facilitar su presentación, después a la imagen se le transformo de RGB a HSV para así poder definir los rangos de color, del más bajo al más alto (color de la piel), con ese rango de valores creamos una máscara y se le guardara la operación And para así solo tomar los valores de color de la piel y por último se le realiza la operación morfológica de apertura para así poder rellenar algunos huecos negros que puede dejar la segmentación.



En esta parte del código se realiza el recorte de donde desea el tatuaje el usuario por medio de la utilización de eventos de mouse para así directamente con el mouse, el usuario pueda dibujar una zona ya sea cuadrada o rectangular en la zona de la piel que el desee. Para así poder guardar como imagen esa parte y poder realizar la suma de imágenes con el dibujo además de también calcular las dimensiones aproximadas del tatuaje.

```
#recorte
xI, yI, xF, yF = 0,0,0,0
interruptor =False

def dibujar(event ,x ,y ,flags ,param):
    global xI,yI,xF,yF,interruptor,imagen
    if event == cv2.EVENT_LBUTTONDOWN:
        xI,yI = x,y
        interruptor =False

    if event == cv2.EVENT_LBUTTONUP:
        xF,yF = x,y
        interruptor =True
        recorte =imagen[yI:yF,xI:xF,:]

        cv2.imwrite("Recorte.png", recorte)

        print("coordenadas" ,"[" ,xI,yI,"]","[" ,xF,yF,"]")
img1 = cv2.imread("p5.jpg")

cv2.namedWindow("Recortar")

cv2.setMouseCallback("Recortar", dibujar)
```



Para comodidad de manejo y de modificación se realiza un pequeño menú “While” el cual preguntara si desea rotar el dibujo para así poder tener una mejor estética en el tatuaje, los valores al cual se puede rotar el dibujo pueden ser positivos en grados (rota a la derecha) y negativos en grados (rota a la izquierda).

```python
rt=1
while rt == 1:
    #rotacion de imagen
    print("Desea rotas la imagen?")
    opc = str(input("si o no?: "))
    if opc=="si" or opc=="SI" or opc=="sI" or opc=="Si":
        print("¿Cuantos grados desea rotar la imagen?")
        angulo = int(input())
        (h, w) = dibujo.shape[:2]
        (cX, cY) = (w // 2, h // 2)

        M = cv2.getRotationMatrix2D((cX, cY), -angulo, 1.0)
        cos = np.abs(M[0, 0])
        sin = np.abs(M[0, 1])

        # calcular las nuevas dimensiones limite de la imagen
        nW = int((h * sin) + (w * cos))
        nH = int((h * cos) + (w * sin))

        # ajustar la matriz de rotación para tener en cuenta la traducción
        M[0, 2] += (nW / 2) - cX
        M[1, 2] += (nH / 2) - cY

        # realizar la rotación real y devolver la imagen
        rotado=cv2.warpAffine(dibujo, M, (nW, nH))
```

La rotación del dibujo se realiza usando el valor del ángulo que el usuario decide escribiendo en la terminal para así poder calcular las nuevas dimensiones de la imagen (Nota la rotación usada, redimensiona la imagen para así no cortar ninguna parte del dibujo)

Después de la rotación del dibujo se le agregara transparencia a la imagen para mejor visualización del tatuaje a la hora se sumar el dibujo con la parte del cuerpo.

```
#Quitar Fondo
image = Image.fromarray(rotado)
rgba = image.convert("RGBA")
datas = rgba.getdata()

newData = []
for item in datas:
    if item[0] == 0 and item[1] == 0 and item[2] == 0:
        #
        newData.append((255, 255, 255, 0))
    else:
        newData.append(item)

rgba.putdata(newData)
rgba.save("dibujo2.png", "PNG")

cv2.imshow("Dibujo rotado",rotado)
cv2.imwrite('dibujo.png', rotado)
```

En la cual consiste en transformar la imagen rotada en RGBA, obtener sus datos para así con un ciclo “for” y la utilización de condicionales encontrar el color negro por su valor RGB y los cambiamos a blanco con transparencia para así quitar el fondo.

```
#-------------------------------------------------------------------------
#suma Imagenes
rs = cv2.imread('recorte.png')
img1 = cv2.imread('dibujo2.png')

invert2 = cv2.bitwise_not(rs)
w=invert2.shape[1]
h=invert2.shape[0]
res = cv2.resize(invert2,(w,h), interpolation=cv2.INTER_CUBIC)
res2 = cv2.resize(img1,(w,h), interpolation=cv2.INTER_CUBIC)
resAW = cv2.subtract(res2,res)
```

Para la suma del dibujo con la parte del cuerpo cargamos la parte del cuerpo que ya habíamos recortado hace unos momentos y el dibujo.
Le aplicamos el operador NOT a la parte recortada y enseguida guardamos su ancho y alto en la variable “w” y “h” respectivamente.
Usando como base las medidas del recorte, redimensionamos las dos imágenes a tamaño iguales para así poder realizas en este caso una resta (debido a que el recorte lo pasamos por una operación Not)

Como se puede notar la imagen del tatuaje tiene un contorno rectangular, eso no ayudará al cálculo de las medidas en centímetros del tatuaje, además, también no servirá para regresar la imagen ya sumada a su lugar en la imagen original.

```
#Muestra final
cv2.imshow('resAW',resAW)
print("tamaño dibujo",resAW.shape)
#Calcular medidas tatuaje
a1=resAW.shape[1]
a2=resAW.shape[0]
ancho=(a1/96)/0.393701
alto=(a2/96)/0.393701
ancho=(ancho*2)-2.5
alto=(alto*2)-2.5
print("---------Tamaño del Tatuaje-----------")
print("Ancho en px: ",a1,"px Ancho en cm: ",round(ancho,2),"cm")
print("Ancho en px: ",a2,"px Alto en cm: ",round(alto,2),"cm")
```

Se guardan en las variables “a1”, “a2” las medidas de la suma del recorte y la imagen, para así poder calcular el ancho y alto del tatuaje usando la conversión de pixeles a centímetros (las imágenes están a 96 puntos por pulgada).
Se guardan las variables “alto” y “ancho” y se les realiza unos últimos cálculos, el *2 es debido a que anterior mente habíamos redimensionado la imagen a la mitad de su tamaño y las restas que se realizan es para acortar el posible error que se tiene al rotar la imagen y redimensionar.

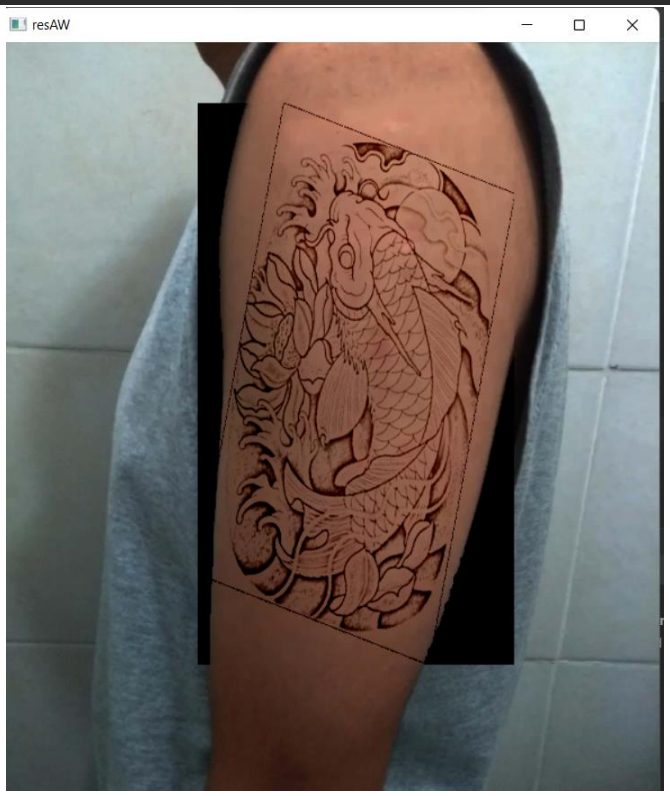
```
tamaño dibujo (674, 381, 3)
---------Tamaño del Tatuaje-----------
Ancho en px:  381 px Ancho en cm:  17.66 cm
Ancho en px:  674 px Alto en cm:  34.17 cm
tamaño imagen (1024, 768, 3)
```

Las ultimas partes de nuestro programa son el regreso de la imagen recortada y ya sumada con el tatuaje a la imagen original

```
    final = cpr
    final2 = cv2.resize(final, None, fx=1/2, fy=1/2, interpolation=cv2.INTER_AREA)
    print("tamaño imagen",final2.shape)
    final2[yI:yF, xI:xF]=resAW
    cv2.imshow('resAW',final2)

    #cv2.imshow('recorte',invert2)
    cv2.waitKey(0)
    cv2.destroyAllWindows()
```

Volvemos a redimensionar la imagen final a su mitad de tamaño para después usando las coordenadas que guardamos cuando el usuario eligió la zona del tatuaje con los eventos de mouse, mandamos a traer el recorte ya con el tatuaje y los guardamos en una nueva variable llamada “resAW” y mostramos.

```
while True:
    img2 = cv2.imread("final.png")
    imagen = cv2.resize(img2, None, fx=1, fy=1, interpolation=cv2.INTER_AREA)
    if interruptor==True:
        cv2.rectangle(imagen,(xI,yI),(xF,yF),(255,0,0),2)
```



Para terminar, mandamos a imprimir las últimas preguntas de nuestro bucle “while” para así poder saber si el usuario está conforme con su diseño o no,

```
    print("Esta conforme con el diseño?")
    print("1)Si , 2)No")
    rt2 = str(input("Digite su respúesta: "))
    if rt2=="2" or rt2=="no" or rt2=="No" :
        rt==1
    if rt2=="1" or rt2=="si" or rt2=="Si":
        break
print("Fin")
```

En caso de estar satisfecho con el diseño el usuario deberá digitar el numero 1 o si en sus diferentes formas como se indica y eso mostrar una impresión de fin.
En caso contrario del que el usuario no este satisfecho, se digitara 2 o la palabra no en sus diferentes formas y la cual mandara al usuario directamente a la pregunta de si desea rotar su imagen para así lograr una mejor estética para el diseño final.

```
---------Tamaño del Tatuaje-----------
Ancho en px:  381 px Ancho en cm:  17.66 cm
Ancho en px:  674 px Alto en cm:  34.17 cm
tamaño imagen (1024, 768, 3)
Esta conforme con el diseño?
1)Si , 2)No
Digite su respúesta: si
Fin
```

Conclusión.

Se logro elaborar un software realmente muy útil, practico y fácil de entender al momento de visualizar el resultado final de una sesión para un tatuaje, este software ayudara tanto al artista encargado de dibujar y hacer el tatuaje como al cliente para que logre ver un resultado final y decida si quiere continuar con el proceso o hacer alguna modificación de tamaño, lugar o diseño del tatuaje.

Este proyecto se realizo en el lenguaje Python y se utilizaron diversas herramientas que fueron vistas en clase como por ejemplo la segmentación de una imagen, sumar dos imágenes y rotarla, al igual que utilizamos funciones nos vistas, por ejemplo, Transparencia de imagen, eventos con el mouse e inversores de color.

El siguiente objetivo de este software poder agregarle mas funciones y que no solamente muestre la implementación de un tatuaje en cualquier zona del cuerpo, uno de los objetivos a futuro es poder eliminar o cubrir algún tatuaje que el usuario ya no desee.