Biometría e identificación: Huella dactilar

Muriel Barreto Juliana, Botello Jesús, Rivera García Mariana
juliana.murielb@autonoma.edu.co
mariana.riveraga@autonoma.edu.co
jesusa.botellop@autonoma.edu.co

***Resumen*— En este informe se presenta el desarrollo del proyecto de biometría e identificación de huellas dactilares desarrollado en la Universidad Autónoma de Manizales en la asignatura de procesamiento digital de imágenes. El cual consiste en el desarrollo de una interfaz gráfica, desarrollada en tkinter, que permite cargar una imagen de una base de datos previamente escogida; a esta imagen se le podrán realizar cuatro operaciones; nitidez, umbralización, brillo y rotación; finalmente se podrá guardar la imagen procesada. Esto será abordado desde la introducción, metodología, resultados y conclusiones.**

Palabras clave: Procesamiento, imagenes, huella dactilar, biometria, operaciones, interfaz gráfica

I. INTRODUCCIÓN

El procesamiento digital de imágenes es una área de la ingeniería que tiene una gran cantidad de aplicaciones en diferentes campos, por ejemplo en la ingeniería biomédica se procesan diferentes imágenes resultante de exámenes médicas, como tomografías. En estos casos se busca mejorar las imágenes para facilitar el diagnóstico de diferentes enfermedades, en ocasiones se busca suprimir algunas siluetas que representan algún órgano o tejido, o en el caso contrario se busca resaltarlas; también se usa el apoyo de algoritmos de entrenamiento para generar sistemas que incluso puedan realizar un diagnóstico sin necesidad de recurrir de forma intermediaria a un médico o especialista de la salúd. En está aplicación se logra apreciar la importancia de estudiar el procesamiento de imágenes, ya que con esto se pueden salvar vidas como en el caso de imágenes médicas y realizar muchas otras cosas más en otros campos.

En este caso en particular decidimos basarnos en el campo de sistemas de identificación y autenticación biométrica, este campo se encarga de procesar imágenes que van a ser usadas en sistemas que comprueban las características biológicas únicas [1], como lo puede ser el iris o la huella dactilar, nuestro proyecto se basará en está última. En la actualidad, cualquier smartphone cuenta con un sistema de reconocimiento de huella dactilar, el cual sirve como sistema de seguridad para acceder al dispositivo. Este mismo mecanismo se puede implementar en una gran variedad de sectores que requieren medios de identificación precisos y una autenticación de usuario rápida, sólida y sencilla: desde las transacciones bancarias, el arranque del coche, el control de acceso a edificios, el cruce de fronteras gubernamentales y, en general, cualquier interacción con las autoridades públicas requiere autenticación [1].

A raíz de la nueva necesidad que surge de crear, mejorar y perfeccionar sistemas de identificación biométricos, se propone la idea de presentar el desarrollo de una interfaz gráfica desarrollada en tkinter, la cual tiene el propósito de mejorar la calidad de imágenes de huellas dactilares, con el fin de que a la hora de implementar estos sistemas mencionados, se puedan obtener mejores resultados en la identificación única de la huella dactilar

II. METODOLOGÍA

Para desarrollar la interfaz gráfica propuesta se realizó una lluvia de ideas de las posibles operaciones a utilizar para mejorar la calidad de las imágenes de huellas dactilares. Previo a este proceso se decide entrar en la búsqueda de una base de datos que nos permita realizar diferentes ensayos con operaciones para procesar las imágenes. El link para acceder a la base de datos se encuentra en la referencia [2].

Tras obtener la base de datos, se analizan las diferentes imágenes de esta, posterior a este análisis se realiza una búsqueda de trabajos previos con objetivos similares al proyecto propuesto; el documento guía principal en el que nos basamos fue el artículo del instituto politécnico nacional en el cual se realiza el procesamiento de una huella dactilar mediante el programa de matlab con el objetivo de realizar una identificación de personas por medio del reconocimiento digital de imágenes biométricas [3].

Posterior a la lectura del artículo mencionado se decide realizar de forma inicial las siguientes operaciones para el procesamiento de las imágenes; la lectura de la imagen, uso de una abertura morfológica para estimar el fondo, sustracción del fondo de la imagen artificial, incrementación del contraste de la imagen, umbralización, etiquetar los objetos de la imagen, examinación de etiquetas de la matriz, observación de las etiquetas, medición de las propiedades de los objetos de la imagen , propiedades estadística y la creación de un histograma para el área de grises.

Además de plantear de forma inicial las operaciones de este artículo, también se propusieron otras, que fueron vistas en la asignatura de procesamiento digital de imágenes, las cuales fueron; la transformada logarítmica, rotación de la imagen, brillo, nitidez, se consideró el filtro mediana para el ruido sal pimienta y finalmente operaciones morfológicas, como la erosión y abertura.

Finalmente se decidieron utilizar 4 operaciones, que fueron las que nos permitieron obtener los mejores resultados a la hora de procesar las imágenes. Fueron la nitidez, la umbralización, el brillo y la rotación; esta última fue incluida porque algunas imágenes no se encontraban de forma centrada. Estas operaciones fueron probadas en la herramienta de Colab, ya que el lenguaje elegido fue el de python.

Cuando ya teníamos los códigos de las operaciones base listos, procedimos a realizar un borrador sobre cómo deseábamos que fuera la interfaz gráfica, a continuación se puede observar el planteamiento inicial de esta.

Primero se propone una pantalla principal, en donde se encuentra el título de la aplicación, el cual fue definido como fingerprint, una imagen y un botón de empezar que llevaría el usuario a la pantalla secundaria.

Imagen 1: Borrador de la pantalla principal de la interfaz

Después se propone que en la pantalla secundaria se pueda acceder a las diferentes imágenes que se encuentran en la base de datos.

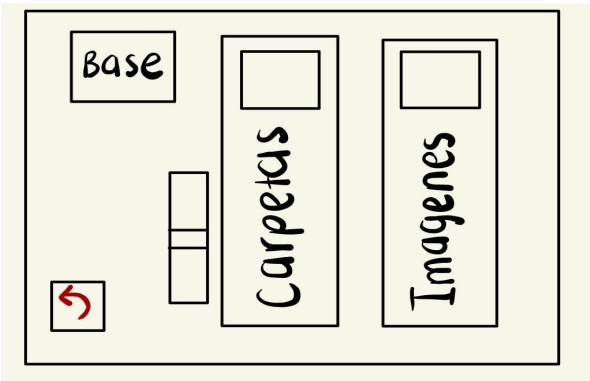

Imagen 2:Borrador segunda pantalla de la interfaz

Finalmente al elegir una imagen se abrirá la tercera pantalla en donde aparecerán botones para efectuar cada una de las operaciones propuestas a la imagen.

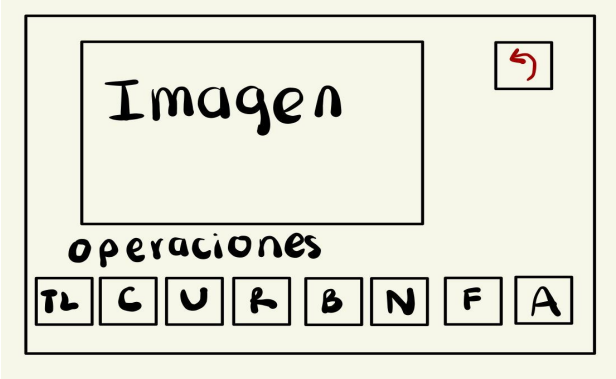

Imagen 3: Borrador tercera pantalla de la interfaz

Tras tener claro como se deseaba que fuera la interfaz y las operaciones a realizar a la imagen, se procede a realizar el código en Tkinter, que es un software que nos permite programar interfaces gráficas en el ambiente de python, este código fue modificado hasta lograr los objetivos planteados de forma inicial.

III. RESULTADOS

La interfaz mostrada en en la imagen 4, se puede observar el inicio de la aplicación en donde se da la bienvenida a los usuarios que van a interactuar con ella.

Imagen 4: inicio de la aplicación

En la imagen 5, se puede observar la aplicación en sí, en dónde está 4 métodos o aplicaciones que se vieron en la materia procesamiento digital de imágenes, estas son: nitidez que se va a encargar de darle más definición o detalle a la imagen, despues esta la umbralización, el cual nos va a permitir distingir la imagen de los objetos del fondo, el cual se encuentra en valores entre 0 y 200, después vamos a encontrar el brillo el cual va a aumentar o disminuir la luminosidad en la imagen, aclarando los colores oscuros y blanqueando los más claros, este se puede modificar entre valores de -200 y 200; finalmente tenemos la rotación la cual nos va a permitir voltear la imágenes que no se encuentren en 180° con el propósito de poder tener una mejor visualización.

imagen 5: Aplicaciones de procesamiento de imágenes en huella dactilar, interfaz gráfica.

A Continuación se presenta el código en sublimeText con el lenguaje de programación de python que se empleó para la realización de la interfaz.

```
from tkinter import *
import tkinter as tk
from tkinter import messagebox
from tkinter import ttk
from tkinter import messagebox
from tkinter import filedialog
from glob import glob
import cv2
from PIL import Image, ImageTk
import numpy as np


pantalla1=Tk()
pantalla2=Toplevel()

pantalla2.withdraw() #cerrar la pantalla 2 cuando se inicialice

#funciones para capturar el valor
'''def actualizar_valor(valor):
    contraste.config(text="Contraste al: " + str(valor))

def actualizar_valor2(valor):
    umbralizacion.config(text="Umbralizacion al: " + str(valor))'''

#FUNCIONES PARA ABRIR Y CERRAR LAS VENTANAS
def abrirVentana1():
    pantalla2.withdraw() #cerrar la pantalla2
    pantalla1.deiconify() #abrir la pantalla1

def abrirVentana2():
    pantalla1.withdraw() #cerrar la pantalla1
    pantalla2.deiconify() #abrir la pantalla2

#-----------------------------------------------------------------
----------------------------------------
#CONFIGURACIONES DE LA PANTALLA 1
pantalla1.title("fingerprint") #titulo del proyecto
pantalla1.resizable(False, False) #para que no se mueva la pantalla
#pantalla1.iconbitmap("Fingerprint-2-icon.ico") icono de la app
pantalla1.geometry("800x550") #dimenciones
pantalla1.config(bg="#EEDEEB") #color fondo de pantalla

# CONFIGURACIONES DEL FRAME 1
miFrame=Frame(pantalla1, width="750", height="500" )
#miFrame.pack(side="left", anchor="n") #ubicacion del frame, expand="true"´para que ocupe toda la pantalla
miFrame.place(relx=0.5, rely=0.5, anchor="center")
miFrame.config(bg="#E4C6DE") #color de fondo del frame
miFrame.config(bd=10) #tamaño del borde
miFrame.config(relief="groove") #borde
miFrame.config(cursor="heart")

#LABEL O TEXTO EN EL FRAME
Label1=Label(miFrame, text=" Bienvenido a fingerprint", fg="#6C4581", font=("Comic Sans MS", 18)).place(x=200, y=6) #TEXTO BIENVENIDO

# IMAGEN EN EL FRAME
miImagen=PhotoImage(file="huella.gif") #cargar una imagen
Label(miFrame, image=miImagen).place(x=160, y=50)

#CONFIGURACIONES DEL BOTON DE LA PANTALLA1
boton=tk.Button(pantalla1, text="Empezar", height=1, width=10, fg="black", font=("Comic Sans MS", 18), command=abrirVentana2) #BOTON DE COMENZAR
boton.place(x=300, y=450)
boton.config(bg="#8D65A3") #color de fondo del frame

#-----------------------------------------------------------------
----------------------------------------------------

#CONFIGURACIONES DE LA PANTALLA 2
pantalla2.title("cargar archivo") #titulo del proyecto
pantalla2.geometry("990x600") #dimenciones
pantalla2.config(bg="#EEDEEB") #color fondo de

#NOMBRES DE LAS FUNCIONES EN PANTALLA 2
nitidez=Label(pantalla2, text=" Nitidez", fg="#6C4581", font=("Comic Sans MS", 13)).place(x=200, y=440)
umbralizacion=Label(pantalla2, text=" Umbralizacion", fg="#6C4581", font=("Comic Sans MS", 13)).place(x=350, y=440)
brillo=Label(pantalla2, text=" Brillo", fg="#6C4581", font=("Comic Sans MS", 13)).place(x=550, y=440)
rotacion=Label(pantalla2, text=" Rotacion",
```

```
fg="#6C4581", font=("Comic Sans MS", 13)).place(x=700, y=440)


# CONFIGURACIONES DEL FRAME 2
miFrame2=Frame(pantalla2, width="375", height="390" )
#miFrame.pack(side="left", anchor="n") #ubicacion del frame, expand="true"´para que ocupe toda la pantalla
miFrame2.place(relx=0.15, rely=0.365, anchor=W)
miFrame2.config(bg="#E4C6DE") #color de fondo del frame
miFrame2.config(bd=10) #tamaño del borde
miFrame2.config(relief="groove") #borde
miFrame2.config(cursor="heart")

# CONFIGURACIONES DEL FRAME 2
miFrame3=Frame(pantalla2, width="375", height="390" )
#miFrame.pack(side="left", anchor="n") #ubicacion del frame, expand="true"´para que ocupe toda la pantalla
miFrame3.place(relx=0.95, rely=0.365, anchor=E)
miFrame3.config(bg="#E4C6DE") #color de fondo del frame
miFrame3.config(bd=10) #tamaño del borde
miFrame3.config(relief="groove") #borde
miFrame3.config(cursor="heart")

#BARRAS HORIZONTALES EN PANTALLA 2
#definicion de rangos para funcion de contraste de 0-100
#definición de rangos para la funcion de umbralizacion 0-200
BarraUmbralizacion = Scale(pantalla2, from_= 0, to=200,orient=HORIZONTAL)
BarraUmbralizacion.place(x=350, y=480)

#BarraNitidez = Scale(pantalla2, orient=HORIZONTAL) #no se usa rangos en la nitidez
#BarraNitidez.place(x=185, y=480) #no se usa rangos en la nitidez

#definicion de rangos para la funcion de brillo de -200 a 200
BarraBrillo = Scale(pantalla2, from_= -200, to=200, orient=HORIZONTAL)
BarraBrillo.place(x=550, y=480)

BarraRotacion = Scale(pantalla2, from_= 0, to=360, orient=HORIZONTAL)
BarraRotacion.place(x=700, y=480)

#FUNCIONES PARA EL MENU EN PANTALLA 2
def salirApp():
    valor=messagebox.askokcancel("salir","estas seguro?")
    if valor==True:
        pantalla1.destroy() #finalizar programa
        pantalla2.destroy() #finalizar programa

 #CARGAR UN ARCHIVO
imagenFrame2 = Label(miFrame2, width="352", height="367") # Crear el Label para mostrar las imagenes de la base de datos
imagenFrame2.place(relx=0.5, rely=0.50, anchor="center")
imagenFrame2.config(bg="white") #color fondo

imagenFrame3 = Label(miFrame3, width="352", height="367") # Crear el Label para mostrar las imagenes de la base de datos
imagenFrame3.place(relx=0.5, rely=0.50, anchor="center")
imagenFrame3.config(bg="white") #color fondo

#CARGAR LA BASE DE DATOS
def abrirArchivo():
    global pos, nombre_imagenes
    directorio = (filedialog.askdirectory())  # Seleccionar una carpeta en el explorador de archivos
    nombre_imagenes = glob(  #sacar lista de archivos en el directorio
        directorio + "/*/*.jpg"
    )  # Retorna una lista con los nombres de archivos
    # en formato png
    #print(nombre_imagenes)
    n_imgs=len(nombre_imagenes)
    pos=0
    miImagen2 = ImageTk.PhotoImage(Image.open(nombre_imagenes[pos]).resize((320,320)))

        #file=nombre_imagenes[0], height=256, width=256
      # Cargar la primera imagen en la carpeta
    imagenFrame2.configure(image=miImagen2)  # Mostrar la imagen
    imagenFrame2.image = miImagen2

def ImgSiguiente():
    global pos, nombre_imagenes
    pos+=1
    miImagen2 = ImageTk.PhotoImage(Image.open(nombre_imagenes[pos]).resize((320,320)))
    #file=nombre_imagenes[0], height=256, width=256 # Cargar la primera imagen en la carpeta
    imagenFrame2.configure(image=miImagen2)  # Mostrar la imagen
    imagenFrame2.image = miImagen2

def ImgAnterior():
    global pos, nombre_imagenes
    pos-=1
    miImagen2 = ImageTk.PhotoImage(Image.open(nombre_imagenes[pos]).resize((320,320)))
        #file=nombre_imagenes[0], height=256, width=256
      # Cargar la primera imagen en la carpeta
    imagenFrame2.configure(image=miImagen2)  # Mostrar la imagen
```

```
    imagenFrame2.image = miImagen2


#leer la imagen de la flecha derecha
image = Image.open("flechaDerecha.png")

#redimensionar la imagen
resize_image = image.resize((30, 50))

#nueva imagen con nuevas dimensiones
img = ImageTk.PhotoImage(resize_image)

#BOTON PARA AVANZAR CON IMAGEN
Bsiguiente=tk.Button(pantalla2, height=30, width=50,
image=img ,justify="right",
fg="black",command=ImgSiguiente)
Bsiguiente.place(x=80, y=80)

#leer la imagen de la flecha izquierda
image2 = Image.open("flechaIzquierda.png")

#redimensionar la imagen
resize_image2 = image2.resize((30, 50))

#nueva imagen con nuevas dimensiones
img2 = ImageTk.PhotoImage(resize_image2)

#BOTON PARA DEVOLVERSE CON IMAGEN
Banterior=tk.Button(pantalla2, height=30, width=50,
image=img2 ,justify="right",
fg="black",command=ImgAnterior)
Banterior.place(x=80, y=150)

#-------------------------------------------------------------------
--------------------------------s
def nitidez():
    global new_img, pos, nombre_imagenes
    coeficientes = np.array([[0, 1, 0],
                             [1, -5, 1],
                             [0, 1, 0]])*-1

img=cv2.imread(nombre_imagenes[pos],cv2.IMREAD_G
RAYSCALE)

#img=cv2.imread(nombre_imagenes[pos],cv2.IMREAD_
GRAYSCALE)
    new_img = cv2.filter2D(img, 0, coeficientes)
    imagen =
ImageTk.PhotoImage(Image.fromarray(new_img).resize((
320,320)))
    imagenFrame3.configure(image=imagen)
    imagenFrame3.image=imagen
    return new_img

bNitidez=tk.Button(pantalla2, text="nitidez", height=1,
width=8, fg="black", font=("Comic Sans MS",
12),command=nitidez)
bNitidez.place(x=200, y=480)

def umbralizacion():
    global new_img, pos, nombre_imagenes
    img=nitidez()

#img=cv2.imread(nombre_imagenes[pos],cv2.IMREAD_
GRAYSCALE)
    ret,new_img =
cv2.threshold(img,BarraUmbralizacion.get(),255,cv2.THR
ESH_BINARY)
    imagen =
ImageTk.PhotoImage(Image.fromarray(new_img).resize((
320,320)))
    imagenFrame3.configure(image=imagen)
    imagenFrame3.image=imagen
    return new_img

bUmbralizacion=tk.Button(pantalla2,
text="umbralizacion", height=1, width=10, fg="black",
font=("Comic Sans MS", 12),command=umbralizacion)
bUmbralizacion.place(x=350, y=530)


def brillo():
    global new_img, pos, nombre_imagenes
    adjustment3 = BarraBrillo.get()
    img=umbralizacion()

#img=cv2.imread(nombre_imagenes[pos],cv2.IMREAD_
GRAYSCALE)
    new_img = np.uint8(np.clip(adjustment3 +
img.astype(np.int16), 0, 255))
    imagen3 =
ImageTk.PhotoImage(Image.fromarray(new_img).resize((
320,320)))
    imagenFrame3.configure(image=imagen3)
    imagenFrame3.image=imagen3
    return new_img

bBrillo=tk.Button(pantalla2, text="brillo", height=1,
width=8, fg="black", font=("Comic Sans MS",
12),command=brillo)
bBrillo.place(x=550, y=530)

def rotacion():
    global new_img, pos, nombre_imagenes

#img=cv2.imread(nombre_imagenes[pos],cv2.IMREAD_
GRAYSCALE)
    img=brillo()
    ancho, alto = img.shape
    center = (ancho/2, alto/2)
    grados= BarraRotacion.get()
    theta = np.deg2rad(grados)
    rotate_matrix = cv2.getRotationMatrix2D(center=center,
angle=grados, scale=1)
```

```
    new_img = cv2.warpAffine(src=img, M=rotate_matrix,
dsize=(ancho, alto))
    imagen =
ImageTk.PhotoImage(Image.fromarray(new_img).resize((
320,320)))
    imagenFrame3.configure(image=imagen)
    imagenFrame3.image=imagen
    return new_img

bRotacion=tk.Button(pantalla2, text="rotacion", height=1,
width=8, fg="black", font=("Comic Sans MS",
12),command=rotacion)
bRotacion.place(x=700, y=530)

#-----------------------------------------------------------------
-----------------------------------------------
#BOTON PARA GUARDAR LA IMAGEN
def guardar():
    global new_img, pos, nombre_imagenes
    #imagen =
ImageTk.PhotoImage(Image.fromarray(new_img).resize((
320,320)))
    imagen=rotacion()
    cv2.imwrite("resultado.png",imagen)

bGuardar=tk.Button(pantalla2, text="Guardar", height=1,
width=8, fg="black", font=("Comic Sans MS",
14),command=guardar)
bGuardar.place(x=850, y=450)
bGuardar.config(bg="#8D65A3") #color de fondo del
frame

#-----------------------------------------------------------------
-----------------------------------------------
#MENU DE LA PANTALLA 2
barraMenu=Menu(pantalla2)
pantalla2.config(menu=barraMenu, width=600,
height=600)

archivoM=Menu(barraMenu, tearoff=0)
archivoM.add_command(label="abrir",
command=abrirArchivo)
archivoM.add_separator()
archivoM.add_command(label="salir",
command=salirApp) #ventana emergente

barraMenu.add_cascade(label="archivo", menu=archivoM)

#CONFIGURACION BOTON DE ATRAS PANTALLA 2
'''# Carga la imagen
imagen = tk.PhotoImage(file="devolver.gif")

# Crea el botón y establece la imagen
boton = tk.Button(pantalla1, image=imagen).place(x=160,
y=50)

# Empaqueta el botón en la ventana
boton.pack()'''

botonAtras=tk.Button(pantalla2, text="Atras", height=1,
width=4, fg="black", font=("Comic Sans MS", 12),
command=abrirVentana1)
botonAtras.place(x=20, y=40)
botonAtras.config(bg="#8D65A3") #color de fondo del

#-----------------------------------------------------------------
-----------------------------------

pantalla1.mainloop()
```

VI. CONCLUSIONES

Gracias a este proyecto logramos entender la cantidad de aplicaciones que pueden tener el procesamiento digital de imágenes, en diversos campos, en este caso en el campo de la biomedicina y adicionalmente a esto logramos desarrollar y entender el propósito de cada una de ellas con éxito.

Estos conocimientos nos permitieron desarrollar una interfaz en el programa sublime Text, en donde aprendimos gran cantidad de herramientas de edición y diseño de interfaces, allí pudimos desarrollar de manera exitosa algunos de los procesamientos que existen para específicamente la biometría e identificación, específicamente en el campo de la huellas dactilar; logramos mejorar y rotar imágenes de este tipo, con el propósito de hacer que la lectura de este sea mucho más sencillo, lo que va a permitir a muchas empresas o personas que trabajen con huellas digitales como métodos de seguridad, que este sea más precisa y su lectura sea mucho más sencilla.

VII. RECONOCIMIENTOS

Profesor Alejandro Mora, mediante estas palabras se expresa total agradecimiento por el conocimiento brindado en las clases de procesamiento digital de imágenes por las asesorías brindadas y para culminar con éxito este proyecto.

VIII. REFERENCIAS

[1] Díaz, G. (2022) *“Identificación Biométrica " tipos, Dispositivos Y Aplicaciones*, *Cinco noticias*. Available at: https://www.cinconoticias.com/identificacion-biometrica/ (Accessed: 16 May 2023).

[2] (No date) Google drive: Sign-in. Available at: https://drive.google.com/ (Accessed: 17 May 2023).

[3] Ortiz, F. (no date) DSpace JSPUI, IPN. Available at: https://tesis.ipn.mx/jspui/ (Accessed: 17 May 2023).