Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

Tipo de artículo: Artículos originales
Temática: Inteligencia artificial
Recibido: 12/07/2024 | Aceptado: 02/09/2024 | Publicado: 30/09/2024

Identificadores persistentes:
DOI: 10.48168/innosoft.s16.a179
ARK: ark:/42411/s16/a179
PURL: 42411/s16/a179

Sistema de control de acceso biométrico mediante reconocimiento facial con técnicas de vivacidad
Biometric access control system through facial recognition with liveness techniques

Kevin Jose Rodriguez Ponce[1][0009-0007-6379-0221]*, **Frank Jhosep Gutierrez Sanchez**[2][0009-0001-5814-1696], **Alberto Carlos Mendoza De los Santos**[3][0000-0002-0469-915X]

[1]Universidad Nacional de Trujillo. Trujillo, Perú. t1013300121@unitru.edu.pe
[2]Universidad Nacional de Trujillo. Trujillo, Perú t1053300521@unitru.edu.pe
[3]Universidad Nacional de Trujillo. Trujillo, Perú amendozad@unitru.edu.pe

*Autor para correspondencia: t1013300121@unitru.edu.pe

Resumen

El presente artículo tiene como finalidad una propuesta de un sistema de reconocimiento facial con técnicas de vivacidad para el control de accesos mediante redes neuronales. El principal enfoque se ha orientado a mejorar la seguridad del acceso un sistema a través de la aplicación de la inteligencia artificial en la biometría evitando cualquier tipo de fraudes y suplantación. Se utilizó Python junto con el gestor de base de datos SQL Server, además del uso de las siguientes bibliotecas como Tkinter, Cv2, Numpy, PIL, Imutils, Mediapipe, Os, Math, Dlib, Face-recognition, Csv, Tkcalendar, Bcrypt y Pyodbc. Los resultados obtenidos al realizar 40 pruebas con una persona real fueron de un acierto de similitud del 82.11 %, por otro lado, para verificar que la persona está realizando un reconocimiento en vivo, se realizaron 20 pruebas con la fotografía de esa persona, en este caso el sistema no permitía el ingreso ya que no se cumplía la verificación de vivacidad, concluyendo así que el sistema de reconocimiento es eficaz y garantiza una mayor seguridad en el control de acceso.

Palabras claves: Control de acceso, Inteligencia artificial, Reconocimiento facial, Técnicas de vivacidad.

Abstract

The purpose of this article is a proposal for a facial recognition system with liveness techniques for access control using neural networks. The main focus has been aimed at improving the security of access to a system through the application of artificial intelligence in biometrics, avoiding any type of fraud and impersonation. Python was used together with the SQL Server database manager, in addition to the use of the following libraries such as Tkinter, Cv2, Numpy, PIL, Imutils, Mediapipe, Os, Math, Dlib, Face-recognition, Csv, Tkcalendar, Bcrypt and Pyodbc. The results obtained by carrying out 40 tests with a real person were a similarity accuracy of 82.11 %, on the other hand, to verify that the person is performing a live recognition, 20 tests were carried out with the photograph of that person, in this case, the system did not allow entry since the liveness verification was not met, thus concluding that the recognition system is effective and guarantees greater security in access control.

Keywords: *Attendace control, Artificial inteligence, Facial recognition, Liveness techniques*

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

Introducción

En el mundo moderno las organizaciones institucionales están cada vez más relacionadas con los servicios de la tecnología, apoyándose de estas para gestionar, mejorar, facilitar sus procesos y actividades tanto de trabajo como académicas para brindar mejores servicios de calidad de educación y poder tener un mejor crecimiento como institución en el ámbito académico y organizacional en el ámbito profesional.

En la actualidad las diversas tecnologías de información y comunicación (TIC), son de gran ayuda para las organizaciones de cualquier ámbito, pero la falta de conocimiento por parte de estas ocasiona flaquezas al momento de implementarlas provocando ciertos errores de inseguridades que pueden afectar, sin que esta se dé cuenta, de manera muy agresiva llegando al punto de hacer colapsar a toda la organización, esto puede darse por diferentes medios pero el más común es mediante el acceso a las páginas institucionales las cuales constan de un login con un nivel de seguridad bastante bajo y es justamente el medio por el cual se suelen dar ataques de ciberdelincuencia [1].

Existen diferentes programas desarrollados que ayudan a cubrir este punto de la inseguridad informática, uno de los que tiene mayor relevancia es el uso de la biometría, esta técnica consiste en el reconocimiento biométrico por el uso de diferentes rasgos anatómicos (como características faciales o dactilares) y de conducta (como la forma de hablar o firmar) [2] para el registro de usuarios los cuales son usados para reconocer su identidad al momento de acceder al sistema de la organización, esto vendría a ser una capa más de seguridad contra los accesos de personas no registradas y/o autorizadas evitando las suplantaciones de identidad.

En este artículo se va a implementar un software el reconocimiento facial con técnicas de vivacidad como la mirada al frente y el parpadeo de ojos, usando así un factor de doble autenticación de tal manera que se tenga una capa más de seguridad y se garantice que la persona es real y no una foto impresa o en un dispositivo electrónico, además, previamente se hará la debida encriptación de la contraseña dentro de la base de datos creada. Una de las múltiples aplicaciones es para los estudiantes de la Universidad Nacional de Trujillo, donde este sistema se puede implementar en el proceso de acceso a su página web llamada SUV2, para lo cual haremos uso de la inteligencia artificial y técnicas de vivacidad para dar mayor de seguridad y porcentaje de efectividad al momento de ser usado.

Materiales y métodos o Metodología computacional

Marco teórico

- **Liveness**: Es un tipo de tecnología que se utiliza en sistemas de seguridad por Biometría en el enfoque del reconocimiento facial esta consiste en la detección no solo de rostros sino de gestos articulados para proteger al usuario de alguna suplantación o falsificación de identidad [3].
- **Inteligencia Artificial**: La Inteligencia Artificial (IA) tiene como objetivo hacer que las máquinas u ordenadores hagan las mismas tareas que pueden ser realizadas por la mente del hombre. Este concepto se conforma de otros dos conceptos como son el Aprendizaje Automático (AA) y el Aprendizaje Profundo (AP) [4].
- **Biometría**: Es una ciencia que estudia las características anatómicas del ser humano basándose en las distancias y posiciones entre las partes del cuerpo con el propósito de poder catalogar y reconoces personas. [5]
- **Ciberseguridad**: La Ciberseguridad hace referencia a un área relacionada con la informática que hace uso de procesos para defender a las máquinas, programas, redes de comunicación, base de datos, servidores y todo lo relacionado con las TIC, de los ataques y accesos no permitidos para poder mantener la confianza de los usuarios [6].
- **Machine Learning**: Es tiene el concepto de Aprendizaje Automático que utiliza las matemáticas, estadísticas y algoritmos informáticos para entrenar a una IA a través de la manipulación de datos [7].
- **Redes Neuronales Convolucionales**: Es un tipo de Red Neuronal Artificial que se caracteriza por tener capas receptivas muy similares a las del cerebro humano. Estas capas son especializadas y clasificadas lo que quiere decir que las capas iniciales detectan características de objetos y se van clasificando según las propiedades hasta llegar a un nivel de clasificación profundo y especializado al punto de poder reconocer objetos (rostros) de una forma exacta [8].

Herramientas y Elementos

- **Python (3.11.5):** es un lenguaje de programación creado por Guido Van Rossum en 1991, es uno de los lenguajes más famosos y utilizados en todo el mundo para la creación de software esto debido a su facilidad de comprensión y uso para construir algoritmos en diferentes áreas [8].
- **Tkinter (8.6):** Paquete de Python que brinda un grupo de herramientas para poder desarrollar y manejar ventanas. [9]

- **CV2 (4.9.0):** Biblioteca de Python de código abierto que contiene algoritmos de visión por computadora para el tratamiento de imágenes [10]
- **Numpy (1.25.2):** Es una librería de Python con funciones y herramientas matemáticas que sirve para procesar y manejar matrices multidimensionales de gran tamaño a gran velocidad [11].
- **PIL (10.3.0):** Biblioteca de Python cuyo nombre es un acrónimo de Python Imaging Library, para el manejo y edición de imágenes. Sostiene formatos de archivos más utilizados como JPEG, GIF y PNG. [12]
- **Imutils (0.5.4):** Es una librería muy usada de Python sustentado en OpenCV que contiene un grupo de funciones para realizar el tratamiento de imágenes como la esqueletización, rotación, cambio de tamaño y entre otras [12]
- **Mediapipe (0.10.14):** Es un proyecto de Google y una librería perteneciente a Python que ofrece soluciones listas y personalizables sobre Machine Learning de código abierto [13].
- **OS:** Es un módulo de Python que proporciona de manera versátil funcionalidades dependientes del sistema operativo para poder interactuar con el [14].
- **Math:** Es un módulo incorporado a la librería de Python que brinda funciones matemáticas estándar [15]
- **Dlib (19.24.1):** Es una librería de Python utilizada para el reconocimiento de caras y conocida especialmente por la detección de 68 puntos clave en el rostro [16]
- **Face-recognition (1.2.3):** Es una biblioteca de Phyton creada por, Adam Geitgey, esta tiene la funcionalidad del reconocimineto facial del dlib, lo cual ayuda bastante facilitando el trabajo. [17]
- **Visual Studio Code:** Editor de código abierto multiplataforma desarrollado por la compañía Microsoft, muy útil por su flexibilidad y versatilidad con multiples lenguajes de programación y ofrece todas las herramientas necesarias de un IDE [18]

Modelo del proceso del sistema de reconocimiento facial con técnicas de vivacidad

En la Figura 1 presentamos el modelo planteado para el control de acceso. Usamos un doble factor de autenticación para garantizar una mayor seguridad, donde el primero es el usuario y contraseña encriptada, luego se reconoce a la persona mirando hacia el frente y el parpadeo 3 veces, de tal manera que así comprobemos que la persona es real.

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

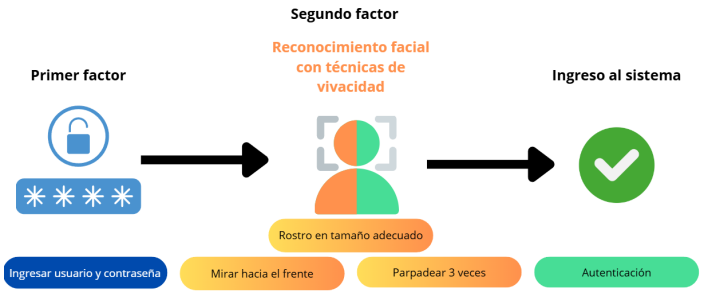

Figura 1. Proceso de ingreso al sistema

Desarrollo del sistema

Dentro de nuestra carpeta general de nuestro proyecto tendremos una subcarpeta llamada ”Rostro”, donde tendremos las caras de los usuarios, el nombre de las fotos en esa carpeta será el ID Usuario, de tal manera que podamos enlazar las fotos con la información que tenemos en nuestra base de datos en SQL Server. Los datos del usuario fueron guardados en la tabla Usuario, donde destacamos el uso de la biblioteca ”bcryp” para el proceso de encriptación de la contraseña como podemos ver en la figura 2.

```
contraseña = RegPass.encode('utf-8')
hashed_password = bcrypt.hashpw(contraseña, bcrypt.gensalt()).decode('utf-8')

try:
    connection_string = (
        'DRIVER={ODBC Driver 13 for SQL Server};'
        'SERVER=.;'
        'DATABASE=django_sqlserver;'
        'Trusted_Connection=yes;'
    )
    conn = pyodbc.connect(connection_string)
    print("Conexión exitosa")
except Exception as ex:
    print(ex)
cursor = conn.cursor()

try:
    cursor.execute("SELECT COUNT(*) FROM Usuarios WHERE IdUsuario = ?", RegUser)
    user_exists = cursor.fetchone()[0] > 0

    if user_exists:
        messagebox.showerror("Error", "El usuario ya está registrado.")
        return

    user_data = {
        'IdUsuario': RegUser,
        'Nombre': RegName,
        'Apellidos': RegLName,
        'Contrasena': hashed_password,
        'Facultad': RegFacultad,
        'Escuela': RegEscuela,
        'FechaNacimiento': RegFechaNac,
        'Genero': RegGenero
    }
```

Figura 2. Código para el registro de los datos del usuario

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

Para la detección de rostros usamos Mediapipe, esta nos permite obtener una malla de rostros con 468 puntos de referencia [19]. Una de las razones por la que la escogimos es por la buena cantidad de puntos faciales que nos ofrece a diferencia de otras bibliotecas, esto nos permite identificar los rostros y obtener los puntos de localización de las partes de la cara para implementarlos en técnicas de vivacidad como mirada al frente, parpadeos, sonrisa, etc. Tenemos un umbral mínimo de detección de rostro definido por defecto como 0.7, además de ello, también un umbral mínimo de similitud que será usado cuando comparemos rostros, en el que es muy importante resaltar que, mientras más cerca esté a cero será más riguroso, debido a que el método para comparar rostros de Face-recognition usa una distancia euclidiana para medir la similitud, por lo que es definido como 0.4 (distancia euclidiana), es decir, en términos más comunes de otras bibliotecas este valor vendría a ser 0.6. Estos valores se definieron por defecto y podrán cambiarse en la interfaz presentada más adelante.

Tenemos la función ”Acceder” en la figura 3, que es una función posterior al ingreso del id usuario y contraseña donde se recibirá como parámetro el ID del usuario que está intentando acceder, donde se ha verificado la contraseña con: ”bcrypt.checkpw(contraseña.encode(), contraseña_hash.encode())”. Luego de darse la autenticación de este primer factor, verificaremos que la carpeta donde se almacenan las caras exista y obtenemos una lista de archivos en esa carpeta, en la línea 654 cargamos la imagen del usuario construyendo la ruta de la imagen del usuario y verificamos su existencia, asimismo, en la línea 659 guardamos la imágen del usuario en la variable ”imgUsuario”, seguidamente en la línea 667 codificamos solo la imagen del rostro del usuario intentando acceder usando la función ”Code_Face” y la guardamos en la variable ”caraGuardada”, la cual será usada más adelante en la comparación de los rostros. Creamos la ventana y llamamos a la función ”Login_Biometrico” donde se realizará el reconocimiento facial usando técnicas de vivacidad.

```
642  def Acceder(usuario):
643      global caraGuardada
644
645      if not os.path.exists(RutaCarpetaCaras):
646          messagebox.showerror("Error", f"La carpeta {RutaCarpetaCaras} no existe.")
647          return
648      lista = os.listdir(RutaCarpetaCaras)
649
650      if not lista:
651          print(f"Error: La carpeta {RutaCarpetaCaras} está vacía.")
652          return
653
654      # Cargamos la imagen del rostro del usuario
655      user_image_path = os.path.join(RutaCarpetaCaras, f"{usuario}.png")
656      if not os.path.exists(user_image_path):
657          messagebox.showerror("Error", f"No se encontró una imagen de rostro para el usuario {usuario}.")
658          return
659
660      imgUsuario = cv2.imread(user_image_path)
661      if imgUsuario is None:
662          messagebox.showerror("Error", f"No se pudo leer la imagen para el usuario {usuario}.")
663          return
664
665      # Codificamos la cara del usuario
666      try:
667          caraGuardada = Code_Face([imgUsuario])
668      except Exception as e:
669          messagebox.showerror("Error", f"Error al codificar la cara: {str(e)}")
670          return
671      crearVentanaAcceder()
672      try:
673          Login_Biometrico(usuario)
674      except Exception as e:
675          messagebox.showerror("Error", f"Error en el login biométrico: {str(e)}")
676          return
```

Figura 3. Código para la lectura y codificación de las imágenes de la carpeta Faces

Ahora, en la figura 4 empezamos el login, para ello, importamos las soluciones de MediaPipe para la detección de

rostros y malla facial. Configuramos ”mp_face_mesh” con los parámetros necesarios, como el número máximo de rostros a detectar y el umbral de confianza mínima que será usado para detectar un rostro. Además, leemos un frame de video y lo convertimos a formato RGB. Luego, procesamos la imagen para detectar puntos faciales usando MediaPipe, en la línea 501 estamos detectando los rostros con ”mallafacial.process”, que había sido definido anteriormente. Luego se detectan puntos faciales, iteramos sobre cada rostro y dibujamos los puntos clave en el frame.

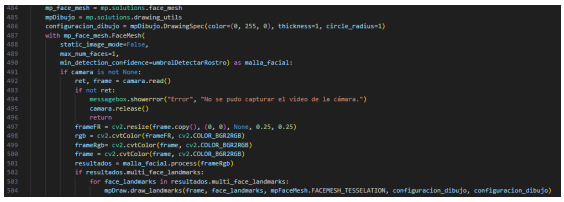

Figura 4. Código para la detección y dibujo de la malla facial

Seguidamente validamos que el rostro esté dentro de unos límites aceptados, es decir que no se encuentre ni muy lejos, ni muy cerca, para este caso se valida que el rostro esté entre 200 y 400 píxeles.

```
def tamaño_cara_adecuado(rect_width, rect_height, landmarks, frame_width, frame_height):
    min_x = min([p['x'] for p in landmarks]) * frame_width
    max_x = max([p['x'] for p in landmarks]) * frame_width
    min_y = min([p['y'] for p in landmarks]) * frame_height
    max_y = max([p['y'] for p in landmarks]) * frame_height

    return (min_x >= 0 and max_x <= frame_width and
            min_y >= 0 and max_y <= frame_height and
            200 < rect_width < 400 and 200 < rect_height < 400)
```

Figura 5. Código para validar el tamaño del rostro

Manejamos la mirada al frente y los parpadeos como técnicas de vivacidad. Inicializamos listas y variables para almacenar las coordenadas de los puntos faciales. identificamos los puntos clave mediante coordenadas, en este caso, detectamos: ojo derecho inferior (145) y superior (159), ojo izquierdo inferior (374) y superior (386), parietal derecho (139), parietal izquierdo (368), ceja derecha (70) y ceja izquierda (300). Los parietales y las cejas son útiles para detectar si una persona está mirando al frente, y para el parpadeo calculamos la longitud entre los puntos de cada ojo.

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

```
longitud1 = math.hypot(puntos_cara[159][0] - puntos_cara[145][0], puntos_cara[159][1] - puntos_cara[145][1])
longitud2 = math.hypot(puntos_cara[386][0] - puntos_cara[374][0], puntos_cara[386][1] - puntos_cara[374][1])

x5, y5 = puntos_cara[139]
x6, y6 = puntos_cara[368]
x7, y7 = puntos_cara[70]
x8, y8 = puntos_cara[300]
```

Figura 6. Código para la detección de puntos clave del rostro

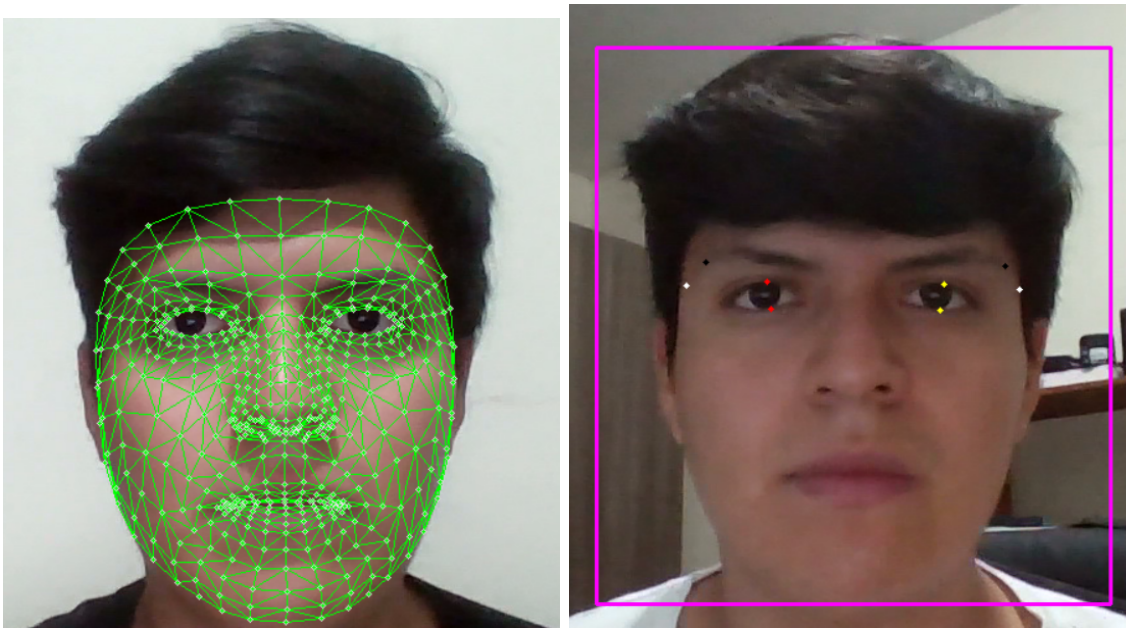

Figura 7. Malla facial de MediaPipe y puntos clave

Ahora verificamos que se esté mirando al frente, x7 y x8 representan la ubicación de la ceja izquierda y derecha respectivamente, mientras que, x5 y x6 los parietales. Entonces, si el valor de la ceja derecha es mayor que el parietal derecho y el valor de la ceja izquierda es mayor que el parietal izquierdo significa que se esté mirando al frente. Si los ojos están cerrados, se incrementa el conteo de parpadeos. Si el conteo de parpadeos es igual a 3 se captura y codifica las caras en ”caras” y ”codificaciones_caras” usando la biblioteca face_recognition, para así llamar a la función ”comparar_rostros” como observamos en la figura 8.

```
caras = fr.face_locations(rgb)
codificaciones_caras = fr.face_encodings(rgb, caras)
comparar_rostros(usuario, codificaciones_caras, umbralComparar, caraGuardada, caraImg)
return
```

Figura 8. Código para codificar los rostros con face_recognition

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

Ahora comparamos el rostro que se quiere evaluar con el rostro almacenado en la carpeta de usuarios, teniendo en cuenta el umbral para comparar rostros inicialmente. Para ello, ”similitudes” almacena la distancia entre las codificaciones faciales usando la biblioteca Face-recognition mediante el método ”fr.face_distance”, este método calcula la distancia euclidiana entre una codificación de rostro y una lista de codificaciones de rostros, que en este caso es el rostro de almacenado del usuario que está intentando ingresar. Cuanto menor sea la distancia, mayor será la similitud entre los rostros. También tenemos ”min_index”, que encuentra el índice de la menor distancia y la variable ”indice_similitud” la almacena. Sí el índice de similitud es menor que el umbral, se autentica al usuario y se llama a muestra el perfil. De lo contrario, se llama a la función ”PerfilNoEncontrad”. En ambos casos se almacena un registro en la tabla Accesos donde se guarda la fecha, hora, id usuario, índice de similitud y el resultado.

```
def comparar_rostros(usuario, codificaciones_caras, umbralComparar, caraGuardada, caraImg):
    for caraEvaluada in codificaciones_caras:
        similitudes = fr.face_distance(caraGuardada, caraEvaluada)
        min_index = np.argmin(similitudes)
        indice_similitud = similitudes[min_index]

        messagebox.showinfo("Completado", "Verificación de vivacidad completada")
        print("SIMILITUD(euclidiana): ", indice_similitud)

        if indice_similitud < umbralComparar:
            registrar_acceso(usuario, indice_similitud, "Acceso concedido")
            Perfil(usuario, indice_similitud, caraImg)
            return
        else:
            registrar_acceso(usuario, indice_similitud, "Acceso denegado")
            PerfilNoEncontrado(usuario, indice_similitud, caraImg)
            return
```

Figura 9. Código para la comparación de la similitud del rostro.

```
def registrar_acceso(usuario, similitud, resultado):
    fecha = datetime.now().strftime("%Y-%m-%d")
    hora = datetime.now().strftime("%H:%M:%S")

    conn_str = (
        'DRIVER={ODBC Driver 13 for SQL Server};'
        'SERVER=.;'
        'DATABASE=BD_ReconocimientoFacial;'
        'Trusted_Connection=yes;'
    )

    try:
        conn = pyodbc.connect(conn_str)
        cursor = conn.cursor()

        cursor.execute("""
            INSERT INTO Accesos (Fecha, Hora, Usuario, Similitud, Resultado)
            VALUES (?, ?, ?, ?, ?)
        """, (fecha, hora, usuario, 1- similitud, resultado))

        conn.commit()
        print("Registro de acceso guardado exitosamente.")

    except Exception as e:
        print(f"Ocurrió un error al guardar el acceso: {e}")

    finally:
        cursor.close()
        conn.close()
```

Figura 10. Código para el almacenamiento de registros de intentos de acceso.

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

Este software de reconocimiento se puede implementar a través de una API en el sistema deseado, sin embargo, diseñamos un prototipo para registrarnos, acceder al sistema y configurar los umbrales:

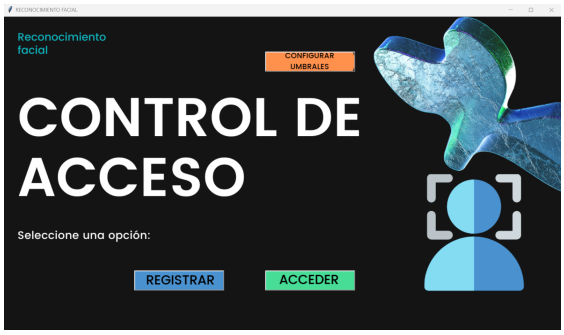

Figura 11. Menú principal del control de acceso

Figura 12. Pantalla de configuración de umbrales

Figura 13. Interfaz para el registro de usuario

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

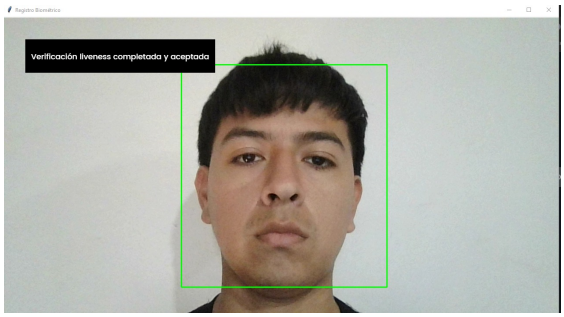

Figura 14. Captura de rostro para el registro de usuario

Figura 15. Formulario de login

Resultados y discusión

En la figura 16 y 17 presentamos una de las pruebas realizadas con el usuario, primero ingresamos las credenciales de usuario y contraseña, para posteriormente hacer la verificación de biometría facial con técnicas de vivacidad. En primer lugar, presentamos el caso de un control aceptado con el usuario registrado previamente, dándonos un porcentaje de similitud de 82.14 %. Para evidenciar la similitud colocamos la imagen guardada del usuario real, sin embargo, en una aplicación final no se debería mostrar.

Revista Innovación y Software
Vol. 5, No. 2, Mes Septiembre - Febrero, 2024
ISSN: 2708-0935
Pág. 114-128
https://revistas.ulasalle.edu.pe/innosoft

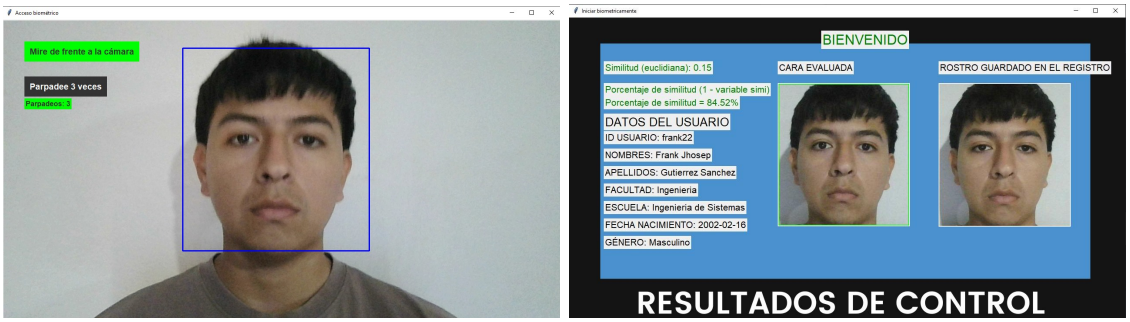

Figura 16. Ejemplo de una de las pruebas de verificación aceptado con su resultado de similitud

En la figura 17 tendremos un usuario externo quiera ingresar al perfil del usuario de la figura 14, mostrando un resultado de similitud de 6,49 % por lo que no se pasa el control.

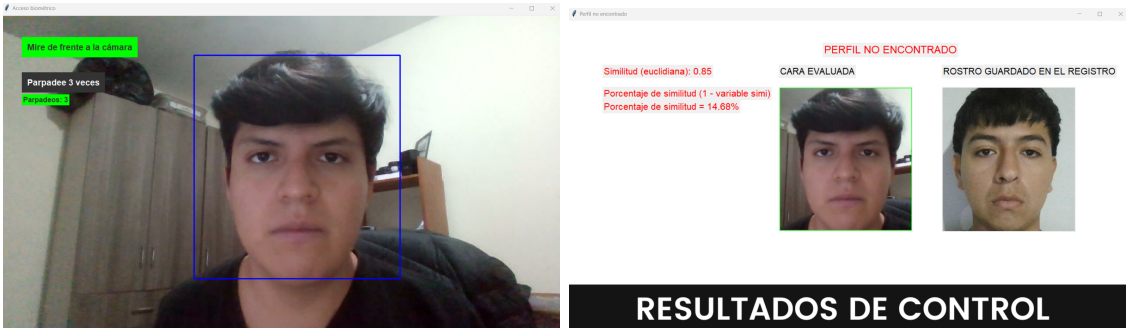

Figura 17. Ejemplo de una de las pruebas de verificación con perfil no encontrado

Luego de hacer 40 predicciones de prueba con el usuario correcto se obtuvo un valor medio de la distancia euclidiana entre los rostros de 0.1788, lo cual indica que el porcentaje de similitud es de un 82,11 %. Es un buen resultado que, en algunos casos debido a la iluminación se puede ver afectado. Con respecto a la mirada al frente y parpadeo, se usó la imagen de la persona proyectada en un celular, si bien es cierto se reconocía la foto como persona, al indicar que se parpadee, no se pudo lograr el acceso de autenticación de persona real. Se realizó posteriormente 19 intentos más, pero no pudimos pasar el control de verificación.

Conclusiones

El avance de la inteligencia artificial es constante y exponencial, su aplicación puede brindar muchos beneficios a los servicios de TI, ya que estos sistemas informáticos pueden desempeñar tareas repetitivas y complejas de

una forma más precisa que los seres humanos o sistemas convencionales.

Con esta propuesta de sistema mejora la seguridad de los accesos de control, al aumentar un factor de autenticación adicional al usuario y contraseña, que a veces pueden interceptarse y usarse malintencionadamente, el reconocimiento facial es útil y más aún aplicamos técnicas de vivacidad para garantizar que la persona es real. Sin embargo, para garantizar aún una mayor seguridad de este sistema se puede implementar técnicas de vivacidad como la detección de texturas, que nos ayudaría a detectar si la persona es real o se está proyectando una foto impresa o en un dispositivo electrónico.

Las bibliotecas que podemos usar en Python para el reconocimiento facial nos ayudan a tener rapidez y confiabilidad en el código, ya que usan redes neuronales convolucionales que involucran algoritmos probados y válidos para detectar rostros y extraer puntos faciales, con resultados rápidos y precisos.

Este sistema de control de acceso se puede implementar en procesos críticos de una organización, por ejemplo, en el Sistema universitario virtual de la Universidad Nacional de Trujillo, donde los docentes podrían autenticarse para publicar calificaciones y los estudiantes para registrar su matrícula. Además, se puede usar en el control de gestos cuando se da un examen de manera virtual, de tal manera que se controlen los movimientos extraños en dicho proceso.

Contribución de Autoría

Kevin Jose Rodriguez Ponce: Conceptualización,Curación de datos, Análisis formal, Investigación, Metodología, Software, Validación, Redacción - borrador original.
Frank Jhosep Gutierrez Sanchez: Conceptualización,Curación de datos, Análisis formal, Investigación, Metodología, Software, Validación, Redacción - borrador original.
Alberto Carlos Mendoza de los Santos: Investigación, Análisis formal, Software, Supervisión, Validación, Redacción - borrador original.

Referencias

[1] N. Juan, M. Ciberdelincuencia, U. Realidad, R. Neira, and J. Manuel, "Universidad piloto de colombia. reyes ciberdelincuencia una realidad - virtual contada a medias," 2024, accedido el 03 de Julio de 2024. [Online]. Available: https://repository.unipiloto.edu.co/bitstream/handle/20.500.12277/2784/Trabajo%20de%20grado.pdf?sequence=1&isAllowed=y

[2] F. Serratosa, "La biometría para la identificación de las personas," 2024, accedido el 03 de

Julio de 2024. [Online]. Available: https://sistemamid.com.ar/panel/uploads/biblioteca/2015-03-22_12-05-01117594.pdf#page=14&zoom=100

[3] S. Chakraborty and D. Das, "An overview of face liveness detection," *arXiv.org*, 2014, accedido el 03 de Junio de 2024. [Online]. Available: https://arxiv.org/abs/1405.2227

[4] A. Pérez del Barrio, P. Menéndez Fernández-Miranda, P. Sanz Bellón, L. Lloret Iglesias, and D. Rodíguez González, "Inteligencia artificial en radiología: introducción a los conceptos más importantes," *Radiología*, vol. 64, no. 3, pp. 228–236, 2022, accedido el 1 de junio de 2024.

[5] F. Serratosa, "La biometría para la identificación de las personas," 2024, accedido el 1 de junio de 2024. [Online]. Available: https://sistemamid.com.ar/panel/uploads/biblioteca/2015-03-22_12-05-01117594.pdf

[6] E. Jove Perez, J. L. Calvo Rolle, D. Urda Muñoz, A. Herrero Cosio, U. Zurutuza, and V. Casola, "Recent advances in the application of data science to industrial cybersecurity," *DYNA*, vol. 96, no. 3, pp. 231–232, 2021.

[7] J. Francisco, R. Veliz, M. Abelardo, and A. Ramírez, "Universidad nacional del callao éstado del arte del aprendizaje automatico relacionado con la logica difusa'," 2024, accedido el 1 de junio de 2024. [Online]. Available: https://repositorio.unac.edu.pe/bitstream/handle/20.500.12952/5580/Informe%20Final-Ramirez%20Veliz-FIIS-2019.pdf?sequence=1&isAllowed=y

[8] Artola, J. Antonio, and P. Carrasco, "Interfaces gráficas de usuario con tk," 2024, accedido el 1 de junio de 2024. [Online]. Available: https://idus.us.es/bitstream/handle/11441/89506/TFG-2402-ARTOLA.pdf?sequence=1&isAllowed=y#page=28&zoom=100

[9] "Interfaces gráficas de usuario con tk," 2024, accedido el 16 de junio de 2024. [Online]. Available: https://docs.python.org/es/3/library/tk.html

[10] "Opencv: Introduction," 2024, accedido el 16 de junio de 2024. [Online]. Available: https://docs.opencv.org/4.x/d1/dfb/intro.html

[11] L. Gonzalez, "Librería numpy - aprende ia," 2020, accedido el 16 de junio de 2024. [Online]. Available: https://aprendeia.com/libreria-de-python-numpy-machine-learning/

[12] L. Chuquimarca Jimenez, S. Pinzon Tituana, and A. Rosales Pincay, "Detección de mascarilla para covid-19 a través de aprendizaje profundo usando opencv y cascade trainer gui," *Revista Científica y Tecnológica UPSE*, vol. 8, no. 1, pp. 68–73, 2021.

[13] X. Teira, N. A. Guerra, G. Castillo, L. Muñoz, and N. González, "Detección de mascarillas utilizando reconocimiento facial," *Tecnología en Marcha*, vol. 36, no. 8, pp. 57–65, 2023.

[14] "os - interfaces misceláneas del sistema operativo - documentación de python - 3.10.13," 2024, accedido el 16 de junio de 2024. [Online]. Available: https://docs.python.org/es/3.10/library/os.html

[15] "math - funciones matemáticas - documentación de python - 3.10.13," 2023, accedido el 16 de junio de 2024. [Online]. Available: https://docs.python.org/es/3.10/library/math.html

[16] A. Tabassum *et al.*, "Drowsiness and distraction detection system using python," 2021, accedido el 16 de junio de 2024. [Online]. Available: https://www.irjmets.com/uploadedfiles/paper/volume3/issue_5_may_2021/11433/1628083464.pdf

[17] A. Rosebrock, "Face recognition with opencv, python, and deep learning," 2018, accedido el 16 de junio de 2024. [Online]. Available: https://pyimagesearch.com/2018/06/18/face-recognition-with-opencv-python-and-deep-learning/

[18] S. Latifi, Ed., *17th International Conference on Information Technology - New Generations (ITNG 2020)*. Cham: Springer International Publishing, 2020, accedido el 16 de junio de 2024.

[19] "Mediapipe face mesh," 2024, accedido el 1 de junio de 2024. [Online]. Available: https://github.com/google-ai-edge/mediapipe/wiki/MediaPipe-Face-Mesh