

Sistema De Reconocimiento Facial Para El Control De Asistencia En Aulas Universitarias

Integrantes:

Líder: Ricardo Andrey Flórez Torres

Julián David Merchán Quiroz

Jorge Iván Monroy Martínez

Alejandro David Contreras Parra

Unidades Tecnológicas de Santander

Tecnología en Desarrollo de Sistemas Informáticos

Docente: Ing. Pedro Ramírez

Bucaramanga, Santander

Fase: **Desarrollo**

1.introducción: En la fase de desarrollo del proyecto “Sistema de Reconocimiento Facial para el Control de Asistencia en Aulas Universitarias”, es esencial seleccionar las herramientas de software adecuadas para asegurar la correcta implementación del sistema. Estas herramientas incluyen programas para la captura y procesamiento de imágenes, algoritmos de reconocimiento facial, almacenamiento de datos, y desarrollo de interfaces de usuario.

2. Programas Seleccionados

2.1. Captura y Procesamiento de Imágenes

Para la fase de captura y procesamiento de imágenes, es necesario utilizar un conjunto de herramientas que permitan la manipulación de imágenes en tiempo real, la detección facial y el pre procesamiento para alimentar el algoritmo de reconocimiento.

Herramientas seleccionadas:

OpenCV (Open Source Computer Vision Library)

Funcionalidades clave:

- Captura de imágenes en tiempo real.
- Detección de rostros en las imágenes.
- Pre procesamiento de imágenes (escalado, normalización).

Razón de selección: OpenCV es ampliamente compatible con lenguajes como Python y C++, lo que facilita su integración en el sistema.

Ejemplo de código:

```
import cv2
import numpy as np

# Cargar el clasificador de Haar para la detección facial
face_cascade = cv2.CascadeClassifier(cv2.data.harcascades + 'haarcascade_frontalface_defa

# Iniciar la captura de video desde la cámara
video_capture = cv2.VideoCapture(0)

while True:
    # Leer un fotograma del video
    ret, frame = video_capture.read()
    if not ret:
        break

    # Convertir el fotograma a escala de grises para mejorar la detección
    gray_frame = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)

    # Detectar rostros en el fotograma
    faces = face_cascade.detectMultiScale(gray_frame, scaleFactor=1.1, minNeighbors=5, min:

    # Dibujar un rectángulo alrededor de cada rostro detectado
    for (x, y, w, h) in faces:
        # Extraer el área del rostro en ↓ fotograma
        face = frame[y:y+h, x:x+w]
```

2.2. Reconocimiento Facial

Para el reconocimiento facial, se utilizarán herramientas y bibliotecas especializadas en la identificación de rostros utilizando algoritmos de machine learning y redes neuronales.

Herramientas seleccionadas:

Dlib

Funcionalidades clave:

- Detección de características faciales clave (ojos, nariz, boca).
- Reconocimiento de rostros basado en modelos entrenados.

Razón de selección: Dlib ha demostrado un rendimiento óptimo para el reconocimiento facial en aplicaciones con restricciones de tiempo real, además de ser de código abierto.



DeepFace (opcional)

Funcionalidades clave:

- Reconocimiento facial con redes neuronales convolucionales (CNN).
- Integración sencilla con bases de datos y sistemas de autenticación.

Razón de selección: Permite experimentar con diferentes modelos para mejorar la precisión del sistema.

Código con la biblioteca Deep face

```
import cv2
from deepface import DeepFace

# Inicializar la cámara
video_capture = cv2.VideoCapture(0)

# Loop para la captura de video
while True:
    # Capturar el fotograma
    ret, frame = video_capture.read()
    if not ret:
        break

    # Intentar analizar el rostro en el fotograma usando DeepFace
    try:
        # El modelo usa el análisis para verificar o identificar rostros
        result = DeepFace.analyze(frame, actions=['age', 'gender', 'race', 'emotion'], enfi

        # Obtener detalles del análisis
        age = result['age']
        gender = result['gender']
        emotion = result['dominant_emoti ↓']
        race = result['dominant race']
```

2.3. Almacenamiento y Gestión de Datos

Herramientas seleccionadas:

MySQL

Funcionalidades clave:

- Almacenamiento de datos biométricos y registros de asistencia.
- Consultas eficientes para obtener la asistencia de los estudiantes.

Razón de selección: MySQL es una opción robusta y ampliamente utilizada en aplicaciones a gran escala, lo que asegura confiabilidad y facilidad de uso en el sistema de asistencia.

Estructura de datos inicial:


```
-- Crear base de datos
CREATE DATABASE IF NOT EXISTS sistema_asistencia;
USE sistema_asistencia;

-- Tabla para almacenar los datos de los estudiantes
CREATE TABLE estudiantes (
    estudiante_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre VARCHAR(100) NOT NULL,
    apellido VARCHAR(100) NOT NULL,
    matricula VARCHAR(20) UNIQUE NOT NULL,
    correo VARCHAR(100),
    foto_perfil BLOB -- Se puede almacenar la foto de perfil para comparación facial
);

-- Tabla para almacenar los cursos
CREATE TABLE cursos (
    curso_id INT AUTO_INCREMENT PRIMARY KEY,
    nombre_curso VARCHAR(100) NOT NULL,
    codigo_curso VARCHAR(20) UNIQUE NOT NULL,
    descripcion TEXT
);

-- Tabla para registrar las sesiones de (↓) curso
```

```
CREATE TABLE sesiones (  
    sesion_id INT AUTO_INCREMENT PRIMARY KEY,  
    curso_id INT NOT NULL,  
    fecha DATETIME NOT NULL,  
    descripcion TEXT,  
    FOREIGN KEY (curso_id) REFERENCES cursos(curso_id)  
);  
  
-- Tabla para registrar la asistencia  
CREATE TABLE asistencias (  
    asistencia_id INT AUTO_INCREMENT PRIMARY KEY,  
    sesion_id INT NOT NULL,  
    estudiante_id INT NOT NULL,  
    hora_asistencia DATETIME NOT NULL,  
    estado ENUM('presente', 'ausente', 'tarde') DEFAULT 'presente',  
    FOREIGN KEY (sesion_id) REFERENCES sesiones(sesion_id),  
    FOREIGN KEY (estudiante_id) REFERENCES estudiantes(estudiante_id)  
);  
  
-- Tabla para almacenar la relación entre estudiantes y cursos  
CREATE TABLE curso_estudiante (  
    curso_estudiante_id INT AUTO_INCREMENT PRIMARY KEY,  
    curso_id INT NOT NULL,  
    estudiante_id INT NOT NULL,  
    FOREIGN KEY (curso_id) REFERENCES cursos(curso_id),  
    FOREIGN KEY (estudiante_id) REFERENCES estudiantes(estudiante_id)  
);
```



SQLite (para pruebas o prototipos)

Funcionalidades clave:

- Almacenamiento de datos en un archivo local sin necesidad de configuración de servidor.

Razón de selección: Se utilizará durante la fase de prototipado para simplificar el desarrollo.


Ejemplo:

```
import sqlite3
from datetime import datetime

# Crear o conectar a una base de datos SQLite
conexion = sqlite3.connect("sistema_asistencia.db")
cursor = conexion.cursor()

# Crear las tablas
cursor.execute('''
CREATE TABLE IF NOT EXISTS estudiantes (
    estudiante_id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre TEXT NOT NULL,
    apellido TEXT NOT NULL,
    matricula TEXT UNIQUE NOT NULL,
    correo TEXT,
    foto_perfil BLOB
)
''')

cursor.execute('''
CREATE TABLE IF NOT EXISTS cursos (
    curso_id INTEGER PRIMARY KEY AUTOINCREMENT,
    nombre_curso TEXT NOT NULL,
    codigo_curso TEXT UNIQUE NOT NULL,
    descripcion TEXT
```



2.4. Desarrollo de la Interfaz de Usuario

HTML: ejemplo de pagina principal que muestra las opciones

```
<!DOCTYPE html>
<html lang="es">
<head>
  <meta charset="UTF-8">
  <title>Sistema de Asistencia</title>
</head>
<body>
  <h1>Sistema de Reconocimiento Facial para Control de Asistencia</h1>
  <ul>
    <li><a href="{{ url_for('registrar_estudiante') }}">Registrar Estudiante</a></li>
    <li><a href="{{ url_for('registrar_curso') }}">Registrar Curso</a></li>
    <li><a href="{{ url_for('consultar_asistencia') }}">Consultar Asistencia</a></li>
  </ul>
  {% with messages = get_flashed_messages() %}
    {% if messages %}
      <ul>
        {% for message in messages %}
          <li>{{ message }}</li>
        {% endfor %}
      </ul>
    {% endif %}
  {% endwith %}
</body>
</html>
```

Fechas previamente establecidas, pueden ser sometidas a cambios o modificaciones

Fase	Septiembre	Octubre	Noviembre	Diciembre
1. Definición de Requerimientos	<div></div>			
2. Diseño de Arquitectura del Sistema	<div></div>			
3. Captura y Procesamiento de Imágenes		<div></div>		
4. Detección y Reconocimiento Facial		<div></div>		
5. Implementación de la Base de Datos			<div></div>	
6. Integración de la Interfaz de Usuario			<div></div>	
7. Pruebas y Optimización				<div></div>
8. Implementación y Ajustes Finales				<div></div>
9. Entrega y Documentación Final				<div></div>