Diseño de un sistema combinacional de biometría facial basado en una red neuronal convolucional para la detección de rostros

Osvaldo Alcázar C., Salvador T. Torres E., Lucero V. Lozano V., Luis E. Avelar G.
oalcazarc1600@alumno.ipn.mx, storrese1502@alumno.ipn.mx, llozanov@ipn.mx, lavelarg@ipn.mx
Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Azcapotzalco, Instituto Politécnico Nacional

Resumen: Se presenta el diseño de un sistema de biometría facial combinando los algoritmos ORB y RANSAC para la identificación de características faciales, haciendo uso de la Red Neuronal Convolucional (CNN) para la detección de rostros obteniendo un porcentaje de efectividad del 94.61% en entornos inestables.

Palabras clave: Reconocimiento facial, red neuronal convolucional, biometría facial.

I) INTRODUCCIÓN

El presente trabajo propone el desarrollo de un sistema de autenticación basado en el reconocimiento facial, con el objetivo de incrementar la seguridad en los sistemas de control de acceso, minimizando las debilidades asociadas a los métodos tradicionales y proporcionando una solución más confiable y eficiente para la protección de información y recursos.

A Planteamiento del problema

La falta de seguridad ante posibles intrusiones, sumada a la creciente demanda de proteger tanto vidas como bienes, constituye un reto importante en aquellos sistemas de seguridad que no cuentan con medidas adicionales de protección. Al depender exclusivamente de la autenticación mediante contraseñas, estos sistemas exponen a los usuarios a un riesgo considerable frente a diversas amenazas. En este sentido, se vuelve imperativo desarrollar un sistema de seguridad más robusto y resistente a falsificaciones, con el fin de asegurar la protección efectiva tanto de individuos como de organizaciones.

B Objetivos

1) Objetivo General

Desarrollar un sistema de biometría facial que permita localizar y reconocer rostros, el cual, puede ser aplicado a diferentes sistemas como por ejemplo sistemas de vigilancia.

2) Objetivos Particulares

Realizar la recopilación de varios algoritmos de reconocimiento facial y la construcción de una base de datos de imágenes, las cuales se emplearán para extraer parámetros específicos que serán utilizados para alimentar un sistema de reconocimiento facial. Para la detección de los rostros, se utilizará el algoritmo YOLO (You Only Look Once), cuyo rol será similar al que tradicionalmente cumple el método de Viola-Jones, permitiendo una identificación rápida y eficaz de los rostros en las imágenes. Una vez detectados los rostros, los algoritmos ORB y RANSAC se encargarán de la detección y emparejamiento de puntos clave de las características faciales, asegurando robustez en el procesamiento de los datos. El sistema será implementado y evaluado con el objetivo de mejorar su capacidad para identificar de manera precisa a los individuos, basándose en las expresiones faciales capturadas, lo que contribuirá a una autenticación más confiable.

II) ANTECEDENTES

A J. Ibarra-Estévez et al [1] presenta el desarrollo de un sistema de reconocimiento facial implementado para control de acceso, el cual utiliza redes neuronales para identificar rostros mediante la extracción de características faciales. La implementación incluye hardware de código abierto, para automatizar el proceso de control de acceso. El sistema fue implantado en una empresa en Ecuador, demostrando ser eficaz en el registro y control de empleados y visitantes, mejorando la seguridad en los entornos laborales.

B I.A. García Torres et al [2] aborda la revisión de tecnologías de reconocimiento facial para desarrollar una API destinada al control y registro de asistencia estudiantil, diseñada para integrarse con diversos sistemas. El estudio incluye un análisis detallado de las tecnologías empleadas previamente en la creación de APIs, así como la evolución del software relacionado. El proyecto hace uso de librerías de código abierto, como OpenCV y Node.js, destacando cómo estas herramientas permiten implementar un sistema eficiente para la gestión de asistencia, mejorando la operación y satisfacción tras su adopción.

C David G. Lowe [3] presenta el desarrollo de un sistema de reconocimiento de objetos utilizando características locales invariantes a la escala. Estas características, conocidas como SIFT (Scale-Invariant Feature Transform), son resistentes a la escala, rotación y parcialmente a los cambios de iluminación, lo que permite el reconocimiento de objetos en entornos complejos y con oclusiones parciales. El enfoque incluye un proceso de verificación por mínimos cuadrados para determinar la validez de las coincidencias y ha mostrado eficacia en imágenes desordenadas y parcialmente ocluidas.

D D. Centurión Talavera et al [4] describe la implementación de un sistema de reconocimiento facial basado en redes neuronales convolucionales (CNN), utilizando una Raspberry Pi 3 como plataforma de procesamiento junto con una cámara para la captura de imágenes. A través del uso de la biblioteca OpenCV y técnicas de aprendizaje profundo, el sistema es capaz de identificar rostros con alta precisión. La investigación resalta que este desarrollo es una alternativa económica y eficiente a los sistemas biométricos convencionales, lo que lo hace adecuado para aplicaciones de seguridad en entornos de bajo costo.

E F. Morcillo Vizuete [5] aborda la creación de un sistema de reconocimiento facial usando técnicas de aprendizaje profundo y la biblioteca OpenCV. Este sistema permite la gestión de usuarios, incluyendo su registro y eliminación, y emplea detección de "liveness" para verificar que interactúa con personas reales, no artefactos. Además, se implementaron pruebas de rendimiento que evaluaron el tiempo de ejecución y la precisión en la identificación de rostros, logrando resultados prometedores en términos de eficiencia operativa.

III) METODOLOGÍA

En esta investigación, se desarrolló un sistema de reconocimiento facial combinando diversas técnicas de procesamiento de imágenes, utilizando MATLAB como herramienta principal para el manejo y análisis de los datos y empleando YOLOv4-Tiny implementado en Darknet [6]. La metodología mezcla conceptos de procesamiento de imágenes, aprendizaje automático y reconocimiento de patrones, abordando fases, desde la adquisición de los datos y el análisis de las imágenes, hasta las pruebas finales para validar el sistema.

A Preparación de la base de datos de imágenes para entrenamiento

Se utilizaron 3098 imágenes obtenidas de la base de datos Wider Face [7], las cuales tiene un formato JPG y diferentes dimensiones pero manteniendo un máximo de altura y/o ancho de 640 pixeles (640×480, 640×360, etc); estas incluyen diversas imágenes donde destacan gesticulaciones y el uso de oclusiones como lentes, gorras o barba, además de incluirse rostros con ligeras inclinaciones. Dichas imágenes, permiten un entrenamiento robusto, generando un aumento en la precisión del sistema de reconocimiento facial propuesto.

B YOLOv4 (You Only Look Once) [8]

Es un algoritmo utilizado para la detección de objetos en tiempo real debido a su eficiencia en velocidad y precisión. A diferencia de los métodos tradicionales que escanean una imagen varias veces (por ejemplo, región por región), YOLO realiza la detección de objetos como una tarea de regresión directa, es decir, que, con una sola pasada por la imagen, predice simultáneamente tanto las clases de los objetos como sus ubicaciones.

1) Arquitectura

Está basada en una red neuronal convolucional (CNN) que procesa toda la imagen de una vez.

- Entrada: La imagen de entrada es de tamaño fijo (generalmente 448×448 píxeles)
- Red Neuronal Convolucional: La imagen se pasa a través de varias capas convolucionales, que extraen características importantes.
- División en Cuadrícula: La imagen se divide en una cuadrícula $S \times S$ y cada celda de la cuadrícula predice B cajas delimitadoras y sus respectivas probabilidades de clase.
- Salidas: Para cada celda, YOLO predice:
  - Coordenadas de las cajas (centro x, y, ancho y alto).
  - Confianza de detección.
  - Clasificación del objeto dentro de la caja.

2) Preparación de datos

Debido a la arquitectura de YOLO, se prepararon los datos para ingresarlos, es decir, se realizó un preprocesamiento de las imágenes en base a los siguientes parámetros:

Width	416
Height	416
Channels	3
Saturation	1.5
Exposure	1.5
Hue	0.1
jitter	0.3
Random	0

Tabla 1: Hiperparámetros para el preprocesamiento de datos

3) Forward Pass (Predicción de Cajas y Clases)

- Predicción de Cajas Delimitadoras (Bounding Boxes)

Para cada celda de la cuadrícula, YOLO predice B cajas delimitadoras y por cada caja, predice las siguientes variables:

(x, y): Coordenadas del centro de la caja delimitadora relativas a la celda de la cuadrícula.

w, h: Ancho y alto de la caja, también relativos al tamaño de la imagen.

C: Confianza de la predicción de la caja, es decir, cuán probable es que la caja contenga un objeto y qué tan precisa es.

$$C = P_{object} = Pr(\text{objeto}) \cdot IOU_{\text{verdadera}}^{\text{predicción}} \quad (1)$$

Donde:

$Pr(\text{objeto})$: Es la probabilidad de que haya un objeto en la celda actual.

$IOU_{\text{verdadera}}^{\text{predicción}}$: Es la superposición entre la caja predicha y la caja real, conocida como Intersección sobre Unión (Intersection over Union, IoU), que mide qué tan bien se ajusta la caja predicha al objeto real.

$$IOU = \frac{\text{Área de intersección entre la caja predicha y la caja real}}{\text{Área de unión de la caja predicha y la caja real}} \quad (2)$$

La confianza será alta si el modelo está seguro de que la caja contiene un objeto, y si la caja predicha se ajusta bien al objeto real (alto IOU).

- Predicción de clases

Cada celda de la cuadrícula también predice la probabilidad de que un objeto pertenezca a una clase específica. Esto se modela como:

$$P(\text{clase}_i|\text{objeto}) = \frac{e^{s_i}}{\sum_j e^{s_j}} \quad (3)$$

Donde:

s_i: Es el puntaje bruto predicho por la red para la clase i de un objeto.

$\sum_j e^{s_j}$: Es la suma de los exponentes de los puntajes para todas las clases predichas, lo que se conoce como función softmax, y transforma los puntajes en probabilidades.

La predicción de clases se realiza para cada celda que contiene un objeto. El resultado es la probabilidad condicional de cada clase dada la presencia de un objeto en la celda.

Se utilizaron los siguientes hiperparámetros para permitir que el modelo realice predicciones sobre la posición de clase de los objetos en la imagen:

Mask	3,4,5 y 0,1,2
Anchors	10,14, 23,27, 37,58, 81,82, 135,169, 344,319
Classes	1
Num	6

Tabla 2: Hiperparámetros para predicciones de posición de clase

- Pérdida (Loss Function)

La función de pérdida utilizada es CIoU (Complete Intersection over Union), la cual está diseñada para optimizar simultáneamente las coordenadas de las cajas delimitadoras, la confianza en la detección y la clasificación de objetos. La función de pérdida total se define de la siguiente manera:

$$L_{\text{total}} = \lambda_{\text{loc}} \cdot L_{CIoU} + \lambda_{\text{conf}} \cdot L_{\text{confianza}} + \lambda_{\text{cls}} \cdot L_{\text{clase}} \quad (4)$$

Donde:

- λ_{loc}: Pondera la importancia de la pérdida de localización, en este caso, la Pérdida CIoU (L_{CIoU})
- λ_{conf}: Pondera la pérdida de confianza
- λ_{cls}: Pondera la pérdida de clasificación.
- L_{CIoU}: Pérdida de CIoU

$$\text{Pérdida CIoU} = 1 - \text{IoU} + \frac{\rho^2(b, b^{gt})}{c^2} + \alpha \cdot v \quad (5)$$

Donde:

IoU*:* Intersección sobre Unión entre la caja predicha y la caja real.

$\rho(b, b^{gt})$: Distancia Euclidiana entre los centros de las cajas predicha (b) y real (b^{gt}).

c: Diagonal del menor cuadro que puede incluir ambas cajas (predicha y real).

v: Término de penalización para la diferencia en la proporción de aspecto entre las cajas.

α: Factor de ajuste para v, que depende de IoU.

Se utilizó iou_normalizer: 0.07 para normalizar la pérdida de IoU para ajustar su impacto en la función de pérdida total.

- $L_{\text{confianza}}$: Pérdida de la confianza en la predicción de la caja. Penaliza tanto los falsos positivos como los falsos negativos:

$$\text{Pérdida de Confianza} = \sum_{i \in \text{cajas}} [y_i \cdot (\widehat{p_i} - 1)^2 + (1 - y_i) \cdot (\widehat{p_i})^2] \quad (6)$$

Donde:

y_i: Etiqueta real, que toma el valor 1 si la caja contiene un objeto y 0 si no lo contiene.

$\widehat{p_i}$: Es la predicción de confianza para la caja i

$y_i \cdot (\widehat{p_i} - 1)^2$: Penaliza cuando la caja debería contener un objeto pero la predicción es baja.

$(1 - y_i) \cdot (\widehat{p_i})^2$: Penaliza cuando la caja no debería contener un objeto pero la predicción de confianza es alta.

- L_{clase}: Pérdida de clasificación, que mide qué tan bien el modelo predice la clase del objeto:

$$\text{Pérdida de Clasificación} = \sum_{i \in \text{cajas con objeto}} \sum_{c=1}^{\text{clases}} \left(\widehat{p_{i,c}} - p_{i,c}\right)^2 \quad (7)$$

Donde:

$\widehat{p_{i,c}}$: Probabilidad predicha de que la caja i pertenezca a la clase c

$p_{i,c}$: Probabilidad real de que la caja i pertenezca a la clase c

En este caso, se utilizó cls_normalizer: 1.0 para normalizar la pérdida de clasificación y ajustar su impacto en la función de pérdida total.

4) Backpropagation

Es un algoritmo que se utiliza para ajustar los pesos en una red neuronal entrenada mediante descenso de gradiente, el cual consiste en propagar el error hacia atrás a través de la red desde las capas finales (salida) hasta las capas iniciales (entrada) para calcular los gradientes con respecto a cada peso en la red, luego, estos gradientes se utilizan para ajustar los pesos en función del error cometido durante el proceso de predicción.

Los gradientes se calculan, como lo indica la *Ecuación 7*, a partir de derivar la función de pérdida con respecto a los pesos, y se propagan a través de la red para ajustar los pesos

5) Optimización (Ajuste de pesos)

Una vez que se propagan los gradientes a través de todas las capas, los pesos de la red se actualizan utilizando estos gradientes para reducir el margen de error en las predicciones, este ajuste se realiza utilizando el algoritmo de optimización Stochastic Gradient Descent (SGD), que es una variante del descenso de gradiente que realiza actualizaciones de los pesos utilizando un subconjunto aleatorio de datos en cada iteración (en lugar de todo el conjunto de datos). SDG se acompaña con

Momentum, que acelera el proceso de entrenamiento al suavizar las actualizaciones de los pesos, acumulando gradientes anteriores para evitar fluctuaciones en las actualizaciones. Mientras que la actualización del gradiente acumulado con Momentum se realiza mediante la *Ecuación 9*, la actualización del peso se calcula con la *Ecuación 10*.

$$v_{t+1} = \beta \cdot v_t + (1 - \beta) \cdot \frac{\partial L}{\partial w_t} \quad (8)$$

$$w_{t+1} = w_t - \alpha \cdot v_{t+1} \quad (9)$$

Donde:

- v_t: Velocidad o gradiente acumulado, que depende de los gradientes calculados en iteraciones previas.
- β: Factor de Momentum, que determina cuánto del gradiente anterior se suma al nuevo gradiente.
- α: Tasa de aprendizaje
- $\frac{\partial L}{\partial w_t}$: Gradiente de la función de pérdida con respecto al peso actual.

Factor de Momentum	0.9
Decay	0.0005
learning_rate	0.00261
burn_in	1000
max_batches	3500
policy	steps
Steps	2800, 3150
Scales	0.1, 0.1

Tabla 3: Hiperparámetros para el ajuste de pesos

6) Evaluación durante el Entrenamiento

El rendimiento de YOLO durante el entrenamiento se evalúa utilizando métricas como:

- Precisión y Recall: Para medir cuántos objetos son detectados correctamente.
- IoU (Intersección sobre Unión): Para medir qué tan bien se ajustan las cajas predichas a los objetos reales.
- mAP (Mean Average Precision): Es una métrica común en detección de objetos que combina precisión y recall en múltiples umbrales de confianza.

7) Repeticiones (Iteraciones) y fin del entrenamiento

Todo el ciclo se repite durante muchas épocas (pasadas completas a través del conjunto de datos de entrenamiento) hasta que el modelo converge y logra un error mínimo razonable.

La cantidad de veces que se necesita repetir el ciclo depende de las épocas y batches (lotes)

Una época es cuando el modelo ha pasado por todo el conjunto de datos de entrenamiento una vez, en cada una, todas las imágenes del conjunto de datos pasan a través del modelo, y se ajustan los pesos del modelo en función del error calculado.

Se definió un batch=64, dividendo cada uno en 16 sublotes (subdivisions) para optimizar la memoria de la GPU

Previamente, se definió el número máximo de iteraciones (max_batches) en 3500, esto significa que el modelo procesará 3500 lotes de datos durante el entrenamiento antes de detenerse.

$$\text{Iteraciones por época} = \frac{\text{Total de imágenes en el conjunto de datos}}{\text{Tamaño del batch}} \quad (10)$$

Ya que, se utilizaron 3098 imágenes de la base de datos Wider Face y el tamaño del lote (batch size) fue de 64, se utilizó la *Ecuación 11* para calcular que se necesitaron aproximadamente 48.4 iteraciones para completar una época, pero como no es posible tener una fracción de una iteración, el último lote tendría menos imágenes. Despejando la ecuación, se concluye que se Se necesitaron aproximadamente. 72.3 épocas para completar las 3500 iteraciones (max_batches).

8) Inferencia (Detección)

Después de que el modelo ha sido entrenado y está listo para hacer predicciones (en fase de inferencia), se repiten las predicciones sobre las cajas delimitadoras, la confianza y las clases, pero esta vez, sobre la imagen de prueba.

9) Supresión de No Máximos (NMS)

Una vez que el modelo ha generado sus predicciones, se usa NMS para eliminar cajas superpuestas y conservar las más relevantes.

confidence_threshold	0.5
ignore_thresh	0.7
truth_thresh	1
IoU Threshold for NMS	0.4
nms_kind: greedynms	Greedynms
beta_nms	0.6

Tabla 4: Hiperparámetros para NMS

Finalmente, se entregan las predicciones finales, que incluyen las coordenadas de las cajas, las clases predichas y la confianza de cada detección.

C ORB (Oriented FAST and Rotated BRIEF) [9] [10]

Es una técnica que combina dos métodos: FAST (Features from Accelerated Segment Test) para la detección de puntos clave y BRIEF (Binary Robust Independent Elementary Features) para la descripción de dichos puntos. Incorpora rotación de características, lo que lo hace resistente a los cambios de orientación en la imagen.

1) Detección de Esquinas con FAST

ORB utiliza el algoritmo FAST para detectar puntos clave o esquinas en una imagen mediante la evaluación de la intensidad de un píxel p y su comparación con los píxeles que lo rodean en un círculo de radio 3. Si un número suficiente de estos píxeles es considerablemente más brillante o más oscuro que el píxel central p. este se clasifica como una esquina, para posteriormente, aplicar supresión de No Máximos y eliminar aquellos puntos clave que no representan la respuesta más fuerte en su vecindario.

2) Cálculo de la orientación de puntos clave: [11]

ORB calcula la orientación de cada punto clave utilizando los momentos de primer orden en su vecindario inmediato, lo que asegura que el descriptor resultante sea invariante a la rotación de la imagen, es decir, el sistema podrá reconocer las características clave del objeto sin importar su orientación en la imagen, lo que garantiza robustez frente a rotaciones.

El ángulo de orientación θ para un punto clave se calcula utilizando los momentos de la imagen:

Cálculo del ángulo de orientación

$$\theta = \arctan\left(\frac{m_{01}}{m_{10}}\right) \quad (11)$$

Donde:

m_{01} y m_{10}: Momentos de primer orden definidos como:

Ecuación para cálculo de m_{01}:

$$m_{01} = \sum_{x,y} y \cdot I(x,y) \quad (12)$$

Ecuación para cálculo de m_{10}:

$$m_{10} = \sum_{x,y} x \cdot I(x,y) \quad (13)$$

Donde:

$I(x,y)$ es la intensidad del píxel en la posición (x,y) dentro de la vecindad del punto clave.

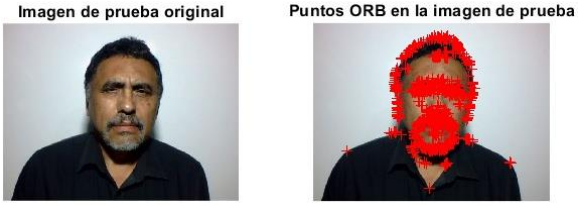

Ilustración 1: Detección y orientación de puntos clave ORB en la imagen de prueba.

3) Rotación de BRIEF (Steered BRIEF)

El descriptor BRIEF es vulnerable a las rotaciones, por lo que ORB rota el descriptor en función del ángulo previamente calculado, para describir los puntos clave. BRIEF funciona comparando la intensidad de pares de píxeles en una vecindad alrededor del punto clave, generando un descriptor binario.

- Ecuación comparativa de rotación de BRIEF

$$fi = \begin{cases} 1 & \text{si } I(p_i) < I(p_j) \\ 0 & \text{si } I(p_i) \geq I(p_j) \end{cases} \quad (14)$$

Donde:

p_i y p_j: Son dos píxeles en la vecindad del punto clave.

4) Creación del Descriptor ORB

El descriptor ORB es una combinación del descriptor BRIEF rotado, generado para cada punto clave; esta rotación es ajustada utilizando la orientación calculada anteriormente.

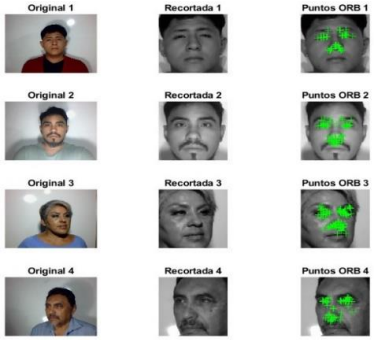

Ilustración 2: Proceso de reconocimiento facial con ORB

5) Emparejamiento de Descriptores

Para permitir una identificación precisa de objetos o patrones en imágenes diferentes, los descriptores ORB generados se comparan con los descriptores de otras imágenes para identificar coincidencias de características. Para medir la similitud entre dos descriptores binarios, se emplea la distancia de Hamming, que cuenta el número de posiciones en las que los bits de los descriptores son diferentes.

La distancia de Hamming d_H entre dos descriptores binarios f_1 y f_2 de longitud n se representa de la siguiente manera:

$$d_H(f_1, f_2) = \sum_{i=1}^{n} \text{XOR}(f_{1i}, f_{2i}) \quad (15)$$

Donde:

XOR: Operación lógica que devuelve 1 si los bits son diferentes, y 0 si son iguales. El resultado es el número de bits diferentes entre los dos descriptores.

D RANSAC (Random Sample Consensus) [12]

Este algoritmo es un método iterativo que busca ajustar un modelo a datos que contienen valores atípicos (outliers), el cual consiste en seleccionar aleatoriamente un subconjunto de datos, ajustar el modelo a este subconjunto, y luego verificar cuántos puntos adicionales (inliers) se ajustan dentro de un umbral predefinido. De este modo, se minimiza la influencia de los outliers, lo que permite que el modelo final refleje mejor los datos principales del conjunto, sin verse afectado significativamente por valores que no siguen el patrón general.

1) Determinación de Inliers

Para todos los puntos restantes en el conjunto de datos, se calcula la distancia perpendicular entre el punto (x_1, y_1) y la línea ajustada con la siguiente función:

Cálculo de distancia perpendicular

$$d_i = |y_i - (mx_i + c)| \quad (16)$$

Si d_i es menor que un umbral ϵ predeterminado, el punto (x_1, y_1) se considera un inlier.

2) Cálculo del número de iteraciones necesarias

$$k = \frac{\log(1-p)}{\log(1-(1-e)^s)} \qquad (17)$$

Donde:

e: Proporción estimada de outliers en los datos.

s: Número mínimo de puntos necesarios para ajustar el modelo (por ejemplo, 2 para una línea).

p: Probabilidad deseada de éxito (que al menos un subconjunto sin outliers sea seleccionado).

3) Cálculo del error total

Es la suma de los errores individuales entre las predicciones del modelo y los valores observados de los inliers, la cual refleja la precisión del ajuste del modelo a los inliers.

$$\text{Error Total} = \sum_{(x_i, y_i) \in \text{inliers}} \left(y_i - (mx_i + c)\right)^2 \qquad (18)$$

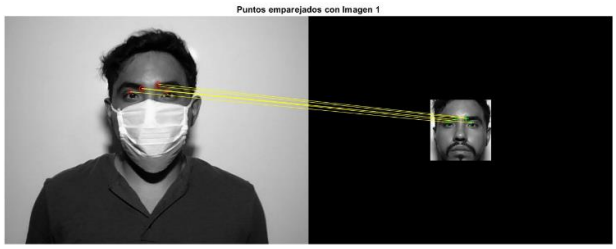

Ilustración 3: Emparejamiento de puntos clave ORB entre la Imagen de prueba y una imagen de la base de datos.

En la Ilustración 3, se observan dos imágenes. La imagen de la izquierda muestra los puntos clave detectados por ORB en la imagen de prueba y filtrados por RANSAC, representados por círculos rojos y en la imagen de la derecha, se identifican los puntos clave con cruces verdes. Las líneas amarillas que conectan los puntos clave entre ambas imágenes indican las correspondencias encontradas. Un mayor número de estas correspondencias sugiere una mayor similitud entre las imágenes, lo que aumenta la probabilidad de que ambas representen el mismo objeto o persona.

E Resultados

El sistema desarrollado para reconocimiento facial utilizó YOLOv4 para la detección de rostro. Durante la fase de pruebas, se evaluó el rendimiento del sistema en un conjunto diverso de imágenes, bajo condiciones de variabilidad de iluminación y orientación, y con varias oclusiones en el fondo de la imagen. En esta fase, se sustituyó YOLO por Viola-Jones [14], para realizar comparativas entre un algoritmo tradicional y uno basado en una red neuronal.

Los resultados indicaron que el sistema es capaz de detectar rostros en diversas condiciones de variabilidad y en tiempo real.

Ilustración 4: Detección de rostros empleando el algoritmo YOLOv4

Posteriormente se emplearon los algoritmos ORB y RANSAC para la detección y emparejamiento de puntos clave y posterior identificación facial.

Los resultados obtenidos muestran una efectividad del 94.61% utilizando YOLO en la identificación y comparación de los rostros frente al 88.89% utilizando Viola-Jones, lo que demuestra la eficacia del sistema propuesto para tareas de reconocimiento facial en tiempo real.

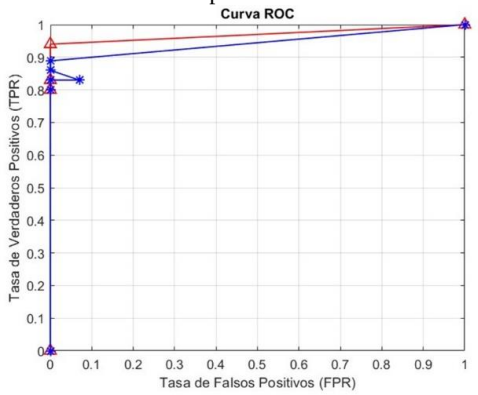

Ilustración 5: Curvas ROC del sistema, en rojo utilizando YOLO para la detección de rostros y en azul utilizando Viola-Jones para el mismo propósito.

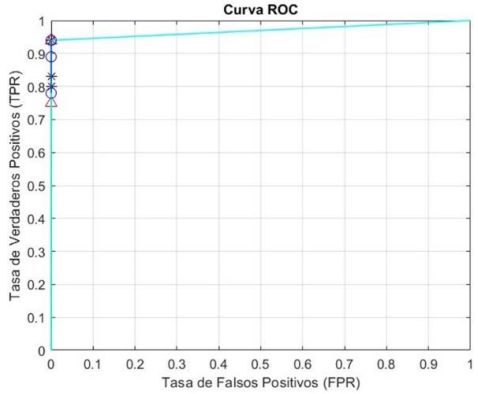

Ilustración 6: Curva ROC del sistema YOLO+ORB+RANSAC. Asterisco: Base de datos propia. Triángulo: Base de datos FERET. Círculo: Base de datos YALE B

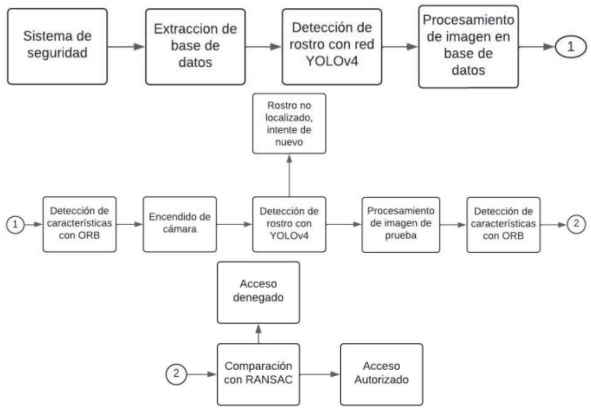

Ilustración 7: Diagrama de flujo del sistema

F Análisis de resultados

1) Viola-Jones + ORB + RANSAC

El sistema de comparación obtuvo una efectividad del 88.89% y buena curva ROC, tuvo algunos falsos positivos, además de un tiempo de detección de 30 a 50 milisegundos.

Viola-Jones + ORB + RANSAC						
No. Persona	TP	FN	FP	TN	TPR	FPR
1	5	1	0	3	0.83	0
2	5	1	1	3	0.83	0.07
3	4	1	0	4	0.80	0
4	6	1	0	2	0.86	0

Tabla 5: Matriz de confusión correspondiente al Reconocimiento mediante Viola-Jones con ORB y RANSAC.

2) YOLO + ORB + RANSAC

Con una efectividad del 94.61% y una curva ROC superior respecto al método con Viola-Jones, sin falsos positivos en las pruebas, y con un tiempo de detección de rostros de 20 a 30 milisegundos, demostró ser más robusto y confiable en términos generales.

RED YOLO+ORB+RANSAC CON BASE DE DATOS PROPIA						
No. Persona	TP	FN	FP	TN	TPR	FPR
1	4	0	0	5	1	0
2	5	1	0	3	0.83	0
3	4	1	0	4	0.8	0
4	4	0	0	5	1	0
5	4	0	0	5	1	0
6	4	1	0	4	0.8	0
7	4	0	0	5	1	0
8	4	1	0	4	0.8	0
9	4	1	0	4	0.8	0
10	4	0	0	5	1	0

Tabla 6: Matriz de confusión correspondiente al Reconocimiento mediante YOLO con ORB y RANSAC con base de datos propia.

RED YOLO+ORB+RANSAC CON BASE DE DATOS FERET						
No. Persona	TP	FN	FP	TN	TPR	FPR
1	3	1	0	3	0.75	0
2	3	1	0	3	0.75	0
3	4	0	0	3	1	0
4	4	0	0	3	1	0
5	4	0	0	3	1	0
6	4	0	0	3	1	0
7	3	1	0	3	0.75	0
8	3	1	0	3	0.75	0
9	4	0	0	3	1	0
10	3	1	0	3	0.75	0
11	4	0	0	3	1	0
12	4	0	0	3	1	0
13	4	0	0	3	1	0

Tabla 7: Matriz de confusión correspondiente al Reconocimiento mediante YOLO con ORB y RANSAC con base de datos FERET

RED YOLO+ORB+RANSAC CON BASE DE DATOS YALE B						
No. Persona	TP	FN	FP	TN	TPR	FPR
1	9	0	0	3	1	0
2	9	0	0	3	1	0
3	9	0	0	3	1	0
4	7	2	0	3	0.78	0
5	8	1	0	3	0.89	0
6	7	2	0	3	0.78	0
7	9	0	0	3	1	0
8	9	0	0	3	1	0
9	7	2	0	3	0.78	0
10	8	1	0	3	0.89	0
11	7	2	0	3	0.78	0
12	8	1	0	3	0.89	0
13	9	0	0	3	1	0
14	9	0	0	3	1	0
15	9	0	0	3	1	0

Tabla 8: Matriz de confusión correspondiente al Reconocimiento mediante YOLO con ORB y RANSAC con base de datos YALE B

G Conclusión

La implementación de un algoritmo que utiliza una red neuronal combinado con los algoritmos robustos de ORB y RANSAC, demuestran una alta capacidad para la identificación de una mayor cantidad de rostros por imagen, un aumento general en la precisión del sistema a 94.61% respecto al 88.89% de Viola-Jones y una reducción en el tiempo de procesamiento, incluso bajo condiciones de variabilidad en iluminación, orientación u oclusión.

H Agradecimientos

Expresamos nuestro más sincero agradecimiento al Instituto Politécnico Nacional y a la Escuela Superior de Ingeniería Mecánica y Eléctrica, Unidad Azcapotzalco, por los recursos, la infraestructura y el ambiente académico proporcionados, así como a la Universidad Nacional Autónoma de México, a la Facultad de Estudios Superiores Cuautitlán, y a todo el personal del Congreso Estudiantil de Inteligencia Artificial Aplicada a La Ingeniería Y Tecnología, (CEIAAIT) por la oportunidad para la presentación de este artículo. Su compromiso con la excelencia y su destacado liderazgo en la promoción de la investigación científica fueron elementos clave que hicieron posible la realización de este proyecto.

IV) REFERENCIAS BIBLIOGRÁFICAS

1. J. Ibarra-Estévez y K. Paredes, "Redes neuronales artificiales para el control de acceso basado en reconocimiento facial", revistapuce, abril de 2018. Accedido el 23 de septiembre de 2024. [En línea]. Disponible: https://doi.org/10.26807/revpuce.v0i106.140

2. I.A. García Torres, X. Trujillo Borja, J. Domínguez De La Torre y W. Navas Espín, "API para control de asistencia con reconocimiento facial mediante OpenCv.JS", RFD, vol. 5, núm. 1 de diciembre de 2021.

3. D. G. Lowe, "Object recognition from local scale-invariant features," Proceedings of the Seventh IEEE International Conference on Computer Vision, Kerkyra, Greece, 1999, pp. 1150-1157 vol.2, doi: 10.1109/ICCV.1999.790410.

4. D. Centurión Talavera y C. Almeida Delgado, «Reconocimiento facial utilizando redes neuronales artificiales en Raspberry Pi», FPUNE Scientific, n.º 16, 2022. [En línea]. Disponible en: http://servicios.fpune.edu.py:83/fpunescientific/index.php/fpunescientific/article/view/234

5. F. Morcillo Vizuete, "Desarrollo de un sistema de reconocimiento facial utilizando Deep Learning con OpenCV", Trabajo de fin de grado, Universitat Politècnica de València, 2019.

6. J. Redmon, «Darknet: Redes neuronales de código abierto en C», 2013-2016.

7. S. Yang, P. Luo, C. C. Loy, y X. Tang, «WIDER FACE: A Face Detection Benchmark», en IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016. [En línea]. Disponible en: http://shuoyang1213.me/WIDERFACE/

8. J. Redmon, S. Divvala, R. Girshick, y A. Farhadi, «You Only Look Once: Unified, Real-Time Object Detection», arXiv preprint arXiv:1506.02640, 2016. [En línea]. Disponible en: https://doi.org/10.48550/arXiv.1506.02640

9. Vinay A., A. S. Rao, V. S. Shekhar, A. K. C., K. N. B. Murthy y S. Natarajan, "Feature extractionusing ORB-RANSAC for face recognition", Procedia Comput. Sci., vol. 70, pp. 174–184, 2015. Accedido el 21 de agosto de 2024. [En línea]. Disponible: https://doi.org/10.1016/j.procs.2015.10.068

10. V. A, A. S. Cholin, A. D. Bhat, K. N. B. Murthy y S. Natarajan, "An efficient ORB based face recognition framework for human-robot interaction", Procedia Comput. Sci., vol. 133, pp. 913–923, 2018. Accedido el 21 de agosto de 2024. [En línea]. Disponible: https://doi.org/10.1016/j.procs.2018.07.095

11. "Detect ORB keypoints - MATLAB detectORBFeatures- MathWorks América Latina". MathWorks - Creadores de MATLAB y Simulink - MATLAB y Simulink - MATLAB & Simulink. Accedido el 19 de agosto de 2024. [En línea]. Disponible: https://la.mathworks.com/help/vision/ref/detectorbfeatures.html

12. R. Krishna, Computer Vision: Foundations and Applications. Stanford, California: Stanford Univ., 2017.

13. P. Viola y M. Jones, "Rapid object detection using a boosted cascade of simple features", en 2001 IEEE Comput. Soc. Conf. Comput. Vis. Pattern Recognit.. CVPR 2001, Kauai, HI, USA. IEEE Comput. Soc, s. f. Accedido el 12 de agosto de 2024. [En línea]. Disponible: https://doi.org/10.1109/cvpr.2001.990517