EDUCACIÓN
SECRETARÍA DE EDUCACIÓN PÚBLICA

TECNOLÓGICO NACIONAL DE MÉXICO

Tecnológico Nacional de México

Centro Nacional de Investigación y Desarrollo Tecnológico

Tesis de Maestría

Sistema de Visión Artificial para el Reconocimiento de la Mascarilla Facial Bien Puesta

presentada por

Ing. Jose Omar de Jesus Trujillo Quintero

como requisito para la obtención del grado de

Maestro en Ciencias de la Computación

Director de tesis

Dra. Andrea Magadán Salazar

Codirector de tesis

Dr. Jonathan Villanueva Tavira

Cuernavaca, Morelos, México. Enero de 2023.

cenidet
Centro Nacional de Investigación y Desarrollo Tecnológico

Centro Nacional de Investigación y Desarrollo Tecnológico
Departamento de Ciencias Computacionales

OFICIO No. DCC/001/2023
Asunto: Aceptación de documento de tesis
CENIDET-AC-004-M14-OFICIO

DR. CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO
PRESENTE

Por este conducto, los integrantes de Comité Tutorial del C. JOSÉ OMAR DE JESÚS TRUJILLO QUINTERO, con número de control M21CE028, de la Maestría en Ciencias de la Computación, le informamos que hemos revisado el trabajo de tesis de grado titulado **"SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO DE LA MASCARILLA FACIAL BIEN PUESTA"** y hemos encontrado que se han atendido todas las observaciones que se le indicaron, por lo que hemos acordado aceptar el documento de tesis y le solicitamos la autorización de impresión definitiva.

Dra. Andrea Magadán Salazar
Directora de tesis

Dr. Jonathan Villanueva Tavira
Codirector de Tesis

Dr. Raúl Pinto Elías
Revisor 1

Dr. Gerardo Reyes Salgado
Revisor 2

C.c.p. Depto. Servicios Escolares.
Expediente / Estudiante

AMS/ibm

Cuernavaca, Mor., 10/enero/2023
No. De Oficio: SAC/006/2023
Asunto: Autorización de impresión de tesis

JOSÉ OMAR DE JESÚS TRUJILLO
CANDIDATO AL GRADO DE MAESTRO EN CIENCIAS DE LA COMPUTACIÓN
P R E S E N T E

Por este conducto, tengo el agrado de comunicarle que el Comité Tutorial asignado a su trabajo de tesis titulado **"SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO DE LA MASCARILLA FACIAL BIEN PUESTA",** ha informado a esta Subdirección Académica, que están de acuerdo con el trabajo presentado. Por lo anterior, se le autoriza a que proceda con la impresión definitiva de su trabajo de tesis.

Esperando que el logro del mismo sea acorde con sus aspiraciones profesionales, reciba un cordial saludo.

A T E N T A M E N T E
Excelencia en Educación Tecnológica®
"Educación Tecnológica al Servicio de México"

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SUBDIRECCIÓN ACADÉMICA

CARLOS MANUEL ASTORGA ZARAGOZA
SUBDIRECTOR ACADÉMICO

C. c. p. Departamento de Ciencias Computacionales
Departamento de Servicios Escolares

CMAZ/RMA

EDUCACIÓN | TECNOLÓGICO NACIONAL DE MÉXICO
10 ENE 2023
CENTRO NACIONAL DE INVESTIGACIÓN Y DESARROLLO TECNOLÓGICO
SERVICIOS ESCOLARES
RECIBIDO
EBN

DEDICATORIA

A mi padre, mi madre, mi hermana y a mi sobrina Alexandra

Por todo su tiempo, apoyo y cariño

AGRADECIMIENTOS

Agradezco al Tecnológico Nacional de México campus Centro Nacional de Investigación y Desarrollo tecnológico (CENIDET) por facilitarme un espacio de trabajo y la oportunidad de crecimiento académico.

Agradezco al Consejo de Nacional de Ciencia y Tecnología (CONACYT) por apoyo económico con la beca que me brindo durante mis estudios de maestría.

Agradezco a Dios por brindarme salud y permitirme cumplir esta meta en mi vida.

Agradezco a mis padres José Trujillo y Alberta Quintero por sus palabras de aliento, tiempo, cariño, motivación y por siempre guiarme por el mejor camino.

Agradezco a mi hermana Juanita y a mi sobrina Alexandra por los momentos de distracción, risas y su cariño incondicional.

Agradezco a la Dra. Andrea por su paciencia, su tiempo, sus consejos y sus enseñanzas.

Agradezco al Dr. Jonathan por siempre alentarme a seguir superándome en mi formación académica, sus comentarios y observaciones.

Agradezco al Dr. José Ruiz, Dr. Raúl Pinto, Dr. Gerardo Reyes por sus cuestionamientos en las revisiones y sus comentarios para la realización de este proyecto de tesis.

Agradezco a la familia Uribe Pérez por recibirme en su hogar como uno más de su familia.

Agradezco a mis compañeros Fernando Luna y Rebecca Zumaya por su colaboración en la creación del conjunto de imágenes del cubrebocas y por su gran amistad.

Agradezco a la secretaria Irma por proporcionarme cubrebocas de diversas tonalidades importantes para el desarrollo de este trabajo.

Finalmente agradezco en general a todos los que directa e indirectamente contribuyeron en la elaboración de este trabajo de tesis.

RESUMEN

Derivado de la pandemia, la Organización Mundial de la Salud (OMS) recomendó utilizar cubrebocas como una forma de contribuir a la disminución de los contagios del virus causante del COVID-19. A pesar de que la pandemia ha disminuido, la OMS recomienda seguir con su uso como medida preventiva para reducir los contagios por la enfermedad y reducir el contagio de otras enfermedades virales. Sin embargo, el uso del cubrebocas provoca que muchos sistemas biométricos faciales actuales sean ineficientes en torno a la localización y reconocimiento de personas al tener el rostro parcialmente ocluido.

En el presente trabajo de tesis se desarrolló un sistema de visión artificial para el reconocimiento de la mascarilla facial bien puesta, es decir, que se encuentre cubriendo la nariz, boca y la barbilla.

El sistema desarrollado considera todas las etapas de un sistema de visión artificial tradicional, permitiendo el ingreso de imágenes desde un archivo, video o mediante el uso de la cámara web de la computadora, realiza la localización del rostro presente en la imagen mediante la herramienta MediaPipe y segmenta tres regiones, donde la plantilla facial utilizada señala como frente, nariz y boca. Se lleva a cabo la descripción de dichas regiones en términos del color (con HSV) y la textura (con LBP), información que ingresa al algoritmo de clasificación Random Forest.

La experimentación realizada muestra que el sistema final es robusto a ambientes reales con un 90.51% de accuracy y 95.01% de F1 score.

ÍNDICE

ÍNDICE DE FIGURAS

ÍNDICE DE TABLAS

ACRÓNIMOS

2D	Dos Dimensiones
3D	Tres Dimensiones
AP	Precisión Media
API	Interfaces de Programación de Aplicaciones
BCOSF	Cascada Mejorada de Características Simples
BSIF	Funciones de Imágenes Estadísticas Binarizadas
CC	Con Lentes-Con Mascarilla
CENIDET	Centro Nacional de Investigación y Desarrollo Tecnológico
CENTRIS	Histograma de Transformación Del Censo
CFP FF	Celebridades de Vista Frontal
CFP FP	Celebridades de Vista Perfil
CMFD	Conjunto de Rostros Correctamente Enmascarados
CNN	Red Neuronal Convolucional
CPU	Unidad Central de Procesamiento
CRC	Clasificación de Representación Colaborativa
CS	Con Lentes-Sin Mascarilla
Dlib	Biblioteca de Software Multiplataforma de Propósito General Escrita en el Lenguaje de Programación C ++
DNN	Red Neuronal Profunda
EER	Tasa de Error Igual
FAR	Tasa de Aceptación Falsa
FDM	Máscara de Descarte de Características
FERET	La Tecnología de Reconocimiento Facial
FFHQ	Flickr-Faces-Hq
FN	Falsos Negativos
FP	Falsos Positivos
FPS	Fotogramas por Segundo
GPU	Unidad de Procesamiento de Gráficos
HOG	Degradado Orientado a Histograma
HSV	H: Tono, S: Saturación, V: Valor
IMFD	Conjunto de Rostros Incorrectamente Enmascarados
IoU	Inserción Sobre Unión
IR	Infra Rojo
JAFFE	Expresiones Faciales Femeninas Japonesas
KNN	Clasificador de Vecinos más Cercanos
LBP	Patrones Binarios Locales
LBPH	Histograma de Patrones Binarios Locales
LDA	Análisis Discriminante Lineal
LFW	Rostros Etiquetados en la Naturaleza

LGC	Codificación de Gradiente Local
LGC-HD	Codificación de Gradiente Local Basada en el Operador Horizontal Y Diagonal
MAFA	Rostros Enmascarados
mAP	Precisión Media Media
MFDD	Detección de Rostro Enmascarado Dataset
MFR2	Reconocimiento de Rostro Enmascarado 2
MTCNN	Redes Convolucionales en Cascada Multitarea
NFS	Sistema de Archivos de Red
NN	Vecino más Cercano
OMS	Organización Mundial de la Salud
ONU	Organización de Las Naciones Unidas
PC	Computadora Personal
PCA	Análisis de Componentes Principales
PDSN	Red Siamesa Diferencial por Pares
PG-CNN	CNN Controlado por Parches
PLOD_FR	Reconocimiento Facial Basado en Detección de Oclusión a Nivel de Píxel
P-SRC	Reconocimiento de Detección Dispersa Particionado
RAM	Memoria de Acceso Aleatorio
RF	Random Forest
RMFRD	Reconocimiento Facial Enmascarado del Mundo Real Dataset
RPCA	Análisis de Componentes Principales Robusto
RPI	Raspberry Pi
SC	Sin Lentes-Con Mascarilla
SIFT	Características Invariantes De Escala
SMFRD	Reconocimiento de Rostro Enmascarado Simulado Dataset
SRC	Representación Dispersa Basada en Clasificación
SS	Sin Lentes-Sin Mascarilla
SSD	Detección de Disparo Único
SVM	Máquina de Vector Soporte
TAR	Tasa de Aceptación Verdadera
TCP/IP	Protocolo de Control de Transmisión/Protocolo de Internet
VGG 16	Red Neuronal Profunda
VGGFACE2	Conjunto de Datos Faciales a Gran Escala
VN	Verdaderos Negativos
VP	Verdaderos Positivos
WGSR	Representación Dispersa Global Ponderada
WIFI	Fidelidad Inalámbrica
YCrCb	Y: Luminancia. Cb, Cr: Características Colorimétricas del Color

Capítulo 1

Introducción

1.1 Introducción

La Organización Mundial de la Salud (OMS) recomienda que todos usen constantemente cubrebocas en público, ya que se ha demostrado que el uso de una máscara facial juega un papel importante en la prevención de la propagación del coronavirus (Organization & others, 2020), debido a que el SARS-COV-2 es transmisible por gotas de saliva y partículas más pequeñas que se expulsan al respirar, toser, estornudar o hablar. La OMS, el 2 de diciembre del 2020 publicó una guía actualizada sobre el uso de mascarillas[1] y en ella se describe en detalle cómo portar el cubrebocas correctamente para garantizar su eficacia. En dicha guía se menciona que la mascarilla debe cubrir completamente la nariz y boca y, se debe ajustar para minimizar cualquier espacio entre el rostro y la mascarilla. Información que se ha difundido entre la población gracias a periódicos[2], hospitales[3] e instructivos[4].

El uso de cubrebocas es común en otros países; sin embargo, en México es algo nuevo para muchas personas. De hecho, a pesar de que ha disminuido el número de contagios de Covid-19 se sigue recomendando su uso para reducir contagios de enfermedades virales que se trasmiten mediante la diseminación de microorganismos que se alojan en la boca, nariz y garganta; sobre todo en espacios cerrados, concurridos y con poca ventilación como el transporte público, las iglesias, centros comerciales, oficinas, elevadores entre otros.

A raíz de la pandemia generada por el COVID-19, los trabajos en torno a localización y reconocimiento de personas considerando la oclusión del rostro se incrementaron, no solo para implementarlos en la búsqueda e identificación de los delincuentes sino para aplicarlos en los sistemas biométricos de acceso; sin

1 La nueva guía de la OMS sobre el uso de mascarillas contra el COVID-19. https://news.un.org/es/story/2020/12/1485002

2 https://www.eleconomista.com.mx/politica/Como-se-usa-el-cubrebocas-20201212-0001.html

3 https://hospitalesangeles.com/covid-19/articulos/Uso-correcto-del-cubrebocas.php

4 https://www.anep.edu.uy/sites/default/files/images/Archivos/publicaciones-direcciones/DSGH/medicos/salud-prevencion/espacio-prevencion/Uso%20correcto%20de%20tapabocas.pdf

embargo, uno de los principales retos es la gran región del rostro ocluida por el cubrebocas.

El desarrollo de sistemas que consideran la oclusión del rostro se ha trabajado durante mucho tiempo como complemento del reconocimiento facial ya que los delincuentes podrían aprovechar de esta debilidad de los sistemas de reconocimiento facial para cometer robos y demás acciones en contra de la ley.

Resumiendo, como contribución a la medida sanitaria se desarrolló un sistema de visión artificial para identificar, de manera automática, si una persona porta mascarilla y ésta se encuentre colocada de forma correcta para permitirle el acceso a sitios públicos y el sistema evita la manipulación humana del equipo salvaguardando la salud de quien pudiera operarlo.

Para ello, el problema principal del sistema es detectar el rostro, parcialmente ocluido de una persona en la imagen analizada y posteriormente, reconocer que el rostro porta una mascarilla en el rostro que cubre la nariz, boca y la barbilla de la persona.

La complejidad del problema es alta debido a que:

- Se requiere llevar a cabo la localización del rostro ocluido ya que los principales elementos que componen el rostro no se tienen a disposición completa por la mascarilla facial; además de la posible presencia de caretas y lentes graduados.

- Existen una gran variedad de colores y uso de figuras en las mascarillas ya que incluso pueden presentar tonos semejantes a la piel humana, e incluso imágenes de la boca logrando un contraste mínimo entre rostro real y el cubrebocas.

- Se puede presentar variabilidad en la iluminación y escala de las imágenes al considerar que se debe utilizar el sistema en un entorno real.

1.2 Problema

El problema es detectar el rostro, parcialmente ocluido de una persona en la imagen analizada y posteriormente, reconocer que la cara porta una mascarilla que cubre la nariz, boca y la barbilla de la persona.

1.3 Objetivos

Objetivo General

Diseñar y desarrollar un sistema que identifique si una persona porta una mascarilla en el rostro y se encuentra puesta de forma correcta, mediante técnicas de visión artificial y aprendizaje automático.

Objetivos Específicos

- Analizar el estado del arte en las áreas de reconocimiento de personas con rostro ocluido y detección de mascarillas en el rostro; además de revisar las técnicas de visión artificial y aprendizaje automático con mejor desempeño.

- Proponer y desarrollar un sistema que identifique cuando una persona porta el cubrebocas cubriendo nariz y boca proporcionando la respuesta en el menor tiempo posible.

- Evaluar el sistema desarrollado con las métricas tradicionales de la clasificación supervisada.

1.4 Alcances y Limitaciones

Alcances

- La imagen a analizar se digitaliza mediante una cámara web o se ingresa al sistema una imagen.

- La imagen contiene el rostro de una persona.

- El sistema realiza el reconocimiento de si una persona porta mascarilla y se encuentra bien colocada; es decir, cubriendo nariz y boca.

- El sistema es capaz de realizar localizar el rostro e identificar la presencia de la mascarilla en el menor tiempo posible.

Limitaciones

- La imagen debe contener la imagen frontal de una persona.
- La mascarilla debe presentar un tono o contrastante respecto al tono de piel de la persona, por ejemplo, azul, blanca y negra.
- El rostro de la persona no debe portar lentes negros u oscuros.
- El sistema considera cambios de iluminación, escala y pequeñas rotaciones.

1.5 Metodología

Para darle solución al problema, se analizó el estado del arte y revisaron técnicas de procesamiento, herramientas de localización y algoritmos de descripción de texturas. Técnicas que fueron implementadas y evaluadas y que permitieron llegar a la metodología de solución que se describe a continuación.

La metodología final propuesta en este trabajo fue estructurada como se muestra en la figura 1.1.

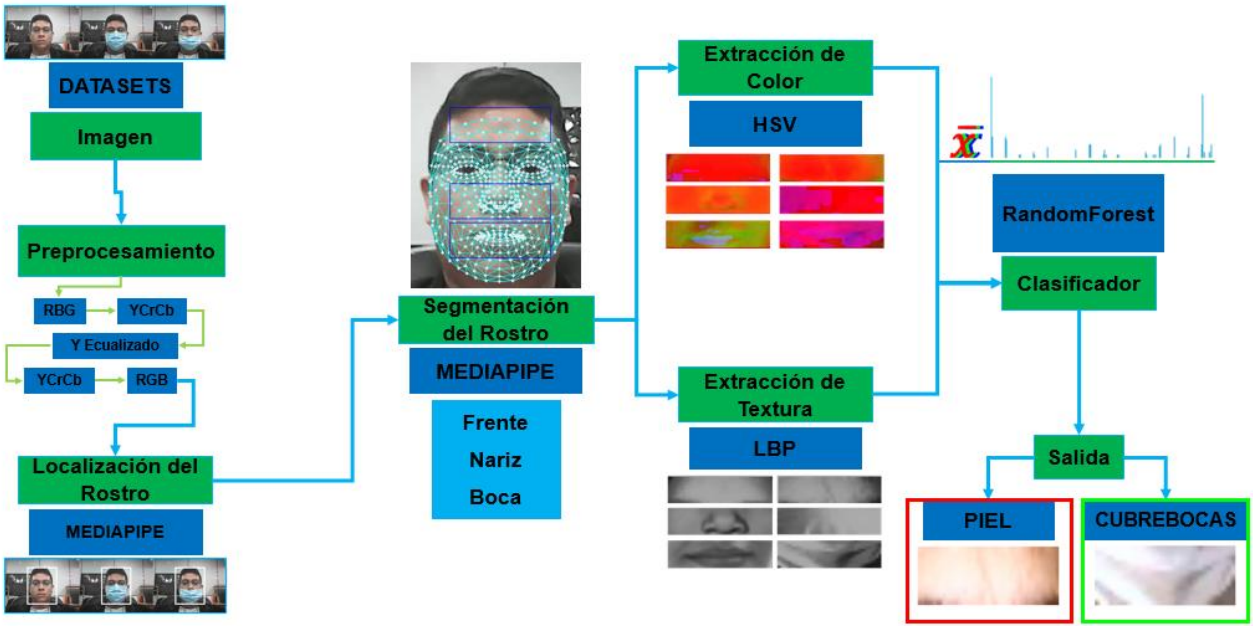


Figura 1.1 Diagrama de flujo de metodología final empleada en este trabajo.

1. Adquisición de la imagen: Se utilizaron conjuntos de imágenes públicas para el entrenamiento y prueba del sistema. Además, se realizó la captura de imágenes

usando una cámara web para ingresarlas en el sistema y comprobar su funcionalidad.

2. Preprocesamiento: Para tratar de minimizar los efectos del cambio de la intensidad luminosa en las imágenes, se optó por transformar la imagen de entrada (que se encuentra en formato RGB) a YCrCb. Se separan los tres canales y se ecualiza el canal *Y* para tener una iluminación uniforme en la imagen. Posteriormente, se unen los tres canales nuevamente y se convierte de nueva cuenta a RGB, obteniendo una imagen RGB con iluminación uniforme.

3. Localización del Rostro: En la etapa de localización, se utilizaron las cuatro herramientas: MediaPipe, Dlib, Detector Facial DNN, MTCNN que no requieren un entrenamiento previo gracias a que cuentan con modelos pre entrenados en la detección del rostro. La herramienta que mejor desempeño tiene es MediaPipe y es la que el sistema utiliza para localizar el rostro presente en la imagen analizada.

4. Segmentación del Rostro: Haciendo uso de la herramienta MediaPipe que cuenta con una malla de 468 puntos de referencia faciales se segmenta el rostro en tres regiones de interés: la frente, nariz y boca.

Para calcular el ancho de las tres regiones del rostro se consideran los puntos correspondientes a las comisuras externas de los ojos, para poder tomar la región sin importar la escala. Con respecto a la altura de cada región, se toman de referencia dos puntos correspondientes a la región a extraer haciendo uso de la malla facial que integra MediaPipe.

Para la región de la frente se ocupan los puntos 9 y 10 con los cuales se calcula la distancia entre ellos, valor que corresponde a la altura del rectángulo de la región ver figura 1.2.

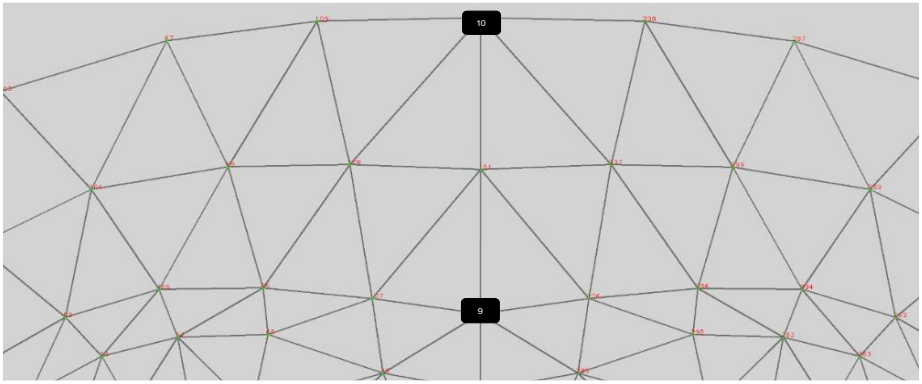


Figura 1.2 Vista de la malla facial región de la frente.

Para obtener la región de la nariz se ocupan los puntos 164 y 195 para obtener el radio y calcular la altura utilizando como punto central el 4, como se observa en la figura 1.3.

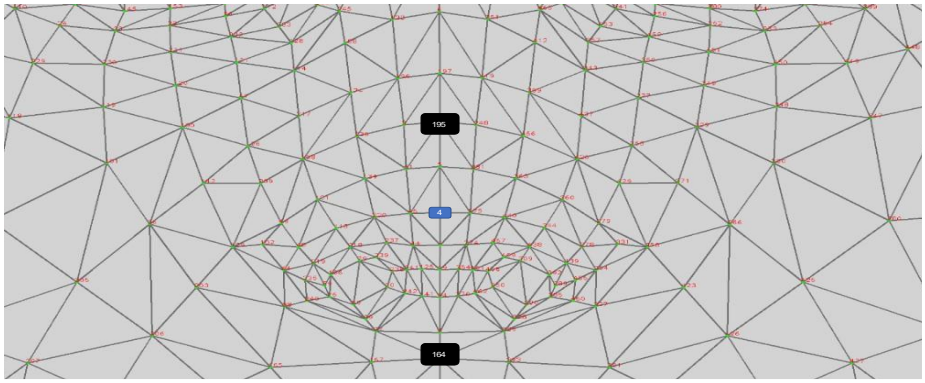


Figura 1.3 Vista de la malla facial región de la nariz.

Finalmente, para la zona de la boca se toman en consideración los puntos 164 y 199 y como punto central entre el 13 y 14, los datos se pueden observar en la figura 1.4.

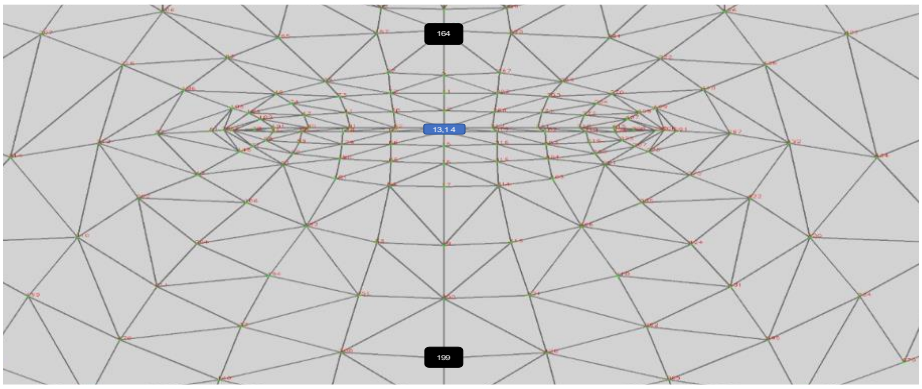


Figura 1.4 Vista de la malla facial región de la boca.

Una vez que se tiene la imagen de cada región, se redimensiona a un tamaño de *150x50* píxeles y cada región se convierte en una imagen individual para su uso posterior.

5. Extracción de Características: Cada una de las regiones obtenida en el paso anterior se describen en términos de color y textura, como se observa en la figura 1.5. Para obtener el descriptor de color, las imágenes resultantes de la segmentación del rostro se transforman de RGB a HSV y se obtiene el promedio de la imagen en sus tres canales, que contribuirá como característica importante entre la detección de la piel y el cubrebocas. Se seleccionó el modelo HSV debido a que el espacio de color está considerablemente más cerca que el espacio de color RGB de la forma en que los humanos experimentan y describen las sensaciones de color (González et al., 2009).

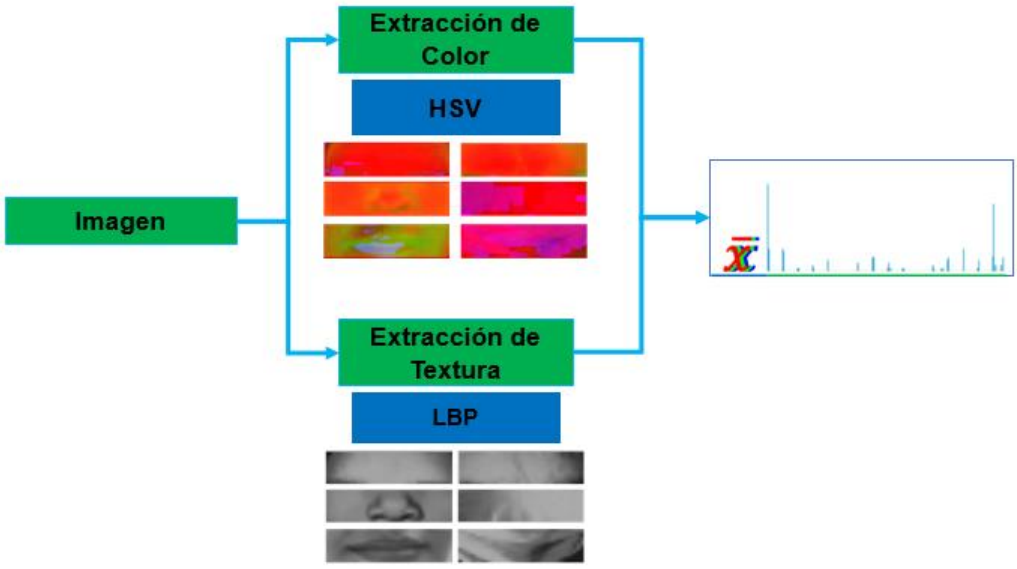


Figura 1.5 Descripción de la región.

Cada una de las tres nuevas imágenes RGB se convierten a escala de grises para extraer los descriptores de textura mediante *Local Binary Patterns* (LBP) (Rosebrock, 2015). LBP se implementó con un *radio=1* y número de *puntos=8*, esta técnica regresa un histograma correspondiente a la descripción de las texturas presentes en la imagen.

El vector de características se crea con el histograma de la textura y el promedio de la imagen HSV y se le asigna la clase a la que pertenecen esos valores, siendo *clase 1* la correspondiente a la piel y *clase 2* la proporcionada para el cubre bocas. Esta información se almacena en un archivo de texto para posteriormente, utilizarla en el entrenamiento y validación de los clasificadores.

6. Clasificación: Se seleccionaron y entrenaron dos clasificadores como se puede apreciar en la figura 1.6. Los clasificadores seleccionados fueron Random Forest y Máquinas de vector soporte (MVS) con la librería sklearn en Python. Con el modelo con mejor desempeño se realiza la predicción con imágenes no consideradas en el aprendizaje y es el establecido en el sistema final. Dichas imágenes se procesan y sólo ingresan al clasificador los vectores de características de las tres regiones ya mencionadas.

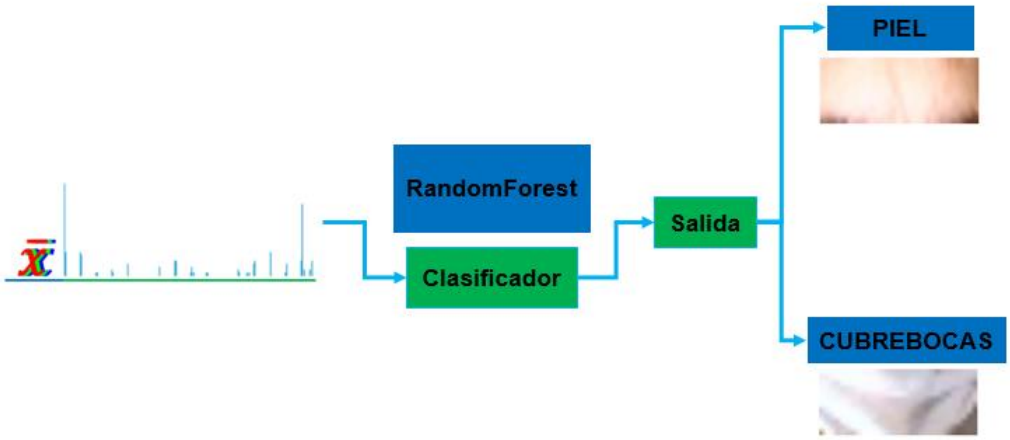


Figura 1.6 Clasificación del vector de características.

7. Salida: El resultado final del sistema se integra de las predicciones de cada una de las regiones. Es decir, se realiza una combinación de los tres valores obtenidos de cada región de interés y haciendo uso de las reglas obtenidas mediante una tabla de verdad se obtiene la respuesta final, como se puede ver en la tabla 1.1, donde *1= a piel* y *2= a cubrebocas*. El color rojo indica la falta del cubrebocas o su uso incorrecto (estado negativo) y el verde señala que el rostro, en la imagen analizada, tiene un cubrebocas puesto de manera correcta (estado positivo).

Tabla 1.1 Tabla de verdad que permite obtener la respuesta final del análisis del rostro. Donde *1=a piel* y *2= a cubrebocas*. El color rojo indica la ausencia del cubrebocas o su uso incorrecto y el verde la presencia de la mascarilla cubriendo nariz y boca.

Frente	Nariz	Boca	Salida
1	1	1	
1	1	2	
1	2	1	
1	2	2	
2	1	1	
2	1	2	
2	2	1	
2	2	2	

La tabla de Verdad se puede resumir en una sola regla:

Si nariz == 2 y boca == 2 entonces es positivo, en caso contrario es negativo.

1.6 Organización de la Tesis

En el Capítulo 1 se habla de la problemática que llevó a desarrollar este proyecto, los objetivos y la metodología propuesta para la solución del problema.

El Capítulo 2 muestra el análisis del estado del arte y los trabajos relacionados llevados a cabo en el TecNM/CENIDET.

El Capítulo 3 presenta el marco teórico de las técnicas y herramientas utilizadas en este trabajo.

El Capítulo 4 contiene la experimentación realizada en este trabajo para cada una de las etapas de la metodología final propuesta; también se realiza la discusión de los resultados obtenidos en este mismo capítulo.

En el Capítulo 5 se proporcionan las conclusiones generales, los logros de los objetivos, las aportaciones, trabajos futuros y las actividades académicas realizadas en el presente trabajo.

En la sección de los anexos se puede encontrar el diseño del sistema, el código implementado y las herramientas utilizadas.

Capítulo 2

Estado del Arte

2.1 Trabajos Relacionados

El presente capítulo contiene la revisión de la literatura referente a la detección del uso de cubrebocas, técnicas de localización del rostro con y sin oclusión, así como trabajos realizados en el TecNM/CENIDET referentes al reconocimiento de expresiones, estados de ánimo y a la identificación de personas.

2.2 Estado del Arte

La sección se integra del análisis de 35 artículos relacionados con la localización y reconocimiento del rostro con oclusión, la detección del cubrebocas en imágenes del rostro para verificar su correcto uso y la creación de conjuntos de imágenes (simuladas y reales) de rostros con cubrebocas.

2.2.1 Reconocimiento de Rostros con Oclusión

A Survey: Face Recognition Techniques Under Partial Occlusion (Azeem et al., 2014)

En el artículo se realizó una recopilación de los principales métodos utilizados en el reconocimiento de rostros en presencia de oclusión; además de detallar los experimentos y los *dataset* utilizados. Se mencionan dos tipos principales de oclusión:

- La oclusión natural: se menciona que se refiere a la obstrucción de vistas entre dos objetos de la imagen sin intenciones
- La oclusión sintética: se menciona que se refiere a cubrir de manera intencionada la vista de la imagen

La oclusión de encuentra presente en diferentes áreas como se puede apreciar en la figura 2.1.

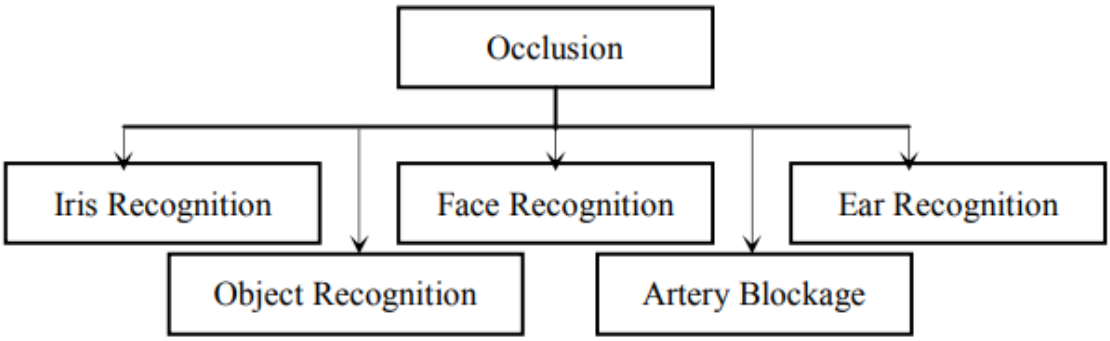


Figura 2.1 Presencia de oclusión en diferentes áreas de procesamiento de imágenes (Azeem et al., 2014).

En el artículo se menciona que se dividen en tres los métodos de reconocimiento facial:

- Métodos basados en características: analizando elementos como ojos, boca, nariz (Sharif et al., 2011) y establecen una correspondencia geométrica entre ellos.
- Métodos basados en piezas: se enfocan en las características holísticas de las imágenes faciales al considerar toda la región de la cara.
- Métodos basados en fractales: se ocupa de las características híbridas locales y globales de las imágenes faciales.

Los métodos anteriores se pueden observar en la figura 2.2 junto con las principales técnicas utilizadas para tratar la oclusión. Los autores mencionan que observaron que los métodos basados en partes, donde la imagen del rostro se dividió en bloques que no se superponen para posteriormente ser analizados en bloques individuales, son los más utilizados. Los resultados obtenidos en el enfoque de reconocimiento facial *2D* se mejoran con técnicas de reconocimiento facial en *3D* para hacer un mejor frente a la oclusión parcial.

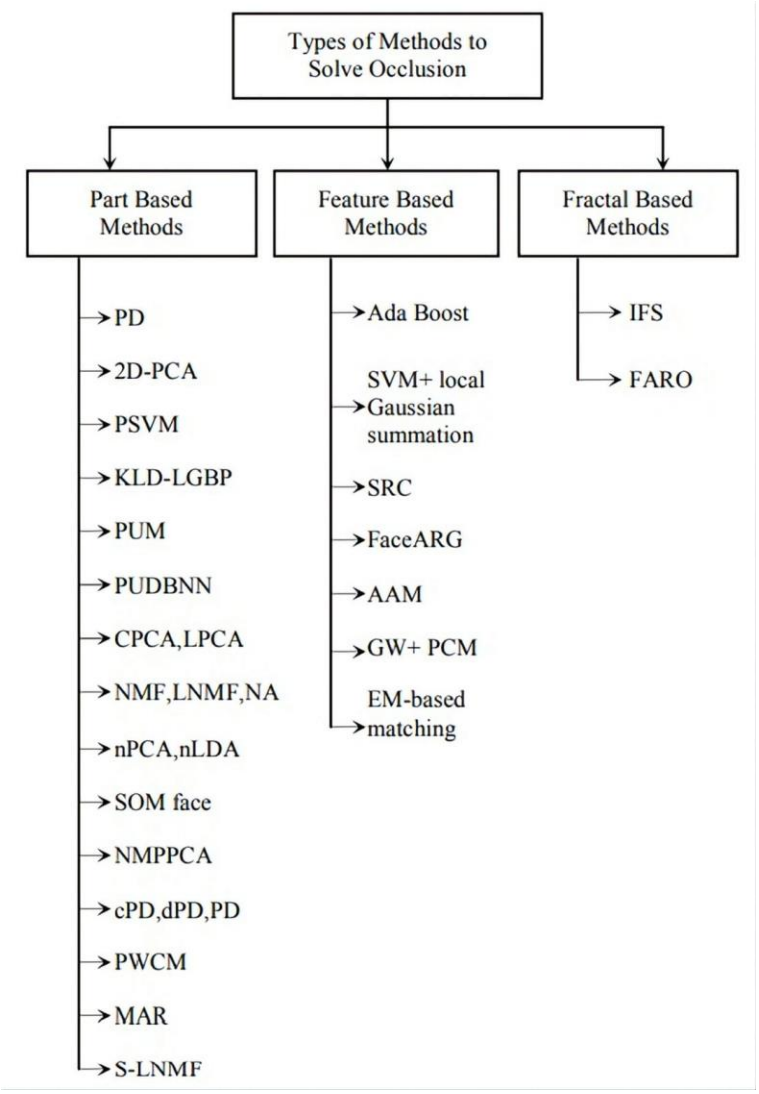


Figura 2.2 Tipos de métodos utilizados para solucionar la oclusión en la imagen facial (Azeem et al., 2014).

Recognition of Occluded Facial Expressions Based on Centrist Features (Cornejo & Pedrini, 2016)

El trabajo presenta el reconocimiento de emociones aun con oclusión facial. Para ello, reconstruyen las regiones ocluidas del rostro aplicando el análisis de componentes principales robusto (RPCA), para posteriormente, extraer los puntos de referencia a través del software CheraFAce and EyesTracking y extraer el conjunto de características de las expresiones faciales, coordenadas del ojo y rotación de los mismos para alinear la imagen. Después, escalan la imagen tomando la distancia más corta entre los ojos y se recorta la región de la cara usando un rectángulo delimitador, y se pasa a escala de grises. Adicionalmente, se agregan rectángulos negros al azar simulando la oclusión, se aplica el descriptor de patrones

binarios locales (LBP) sobre toda la imagen, se aplica el histograma de transformación del censo (CENTRIS) y se reduce la dimensionalidad de las características extraídas mediante el análisis de componentes principales (PCA) y el análisis discriminante lineal (LDA) y se implementa el clasificador de vecinos más cercanos (KNN) y la máquina de vector soporte (SVM) para la clasificación.

Dentro de los *datasets* ocupados se encuentra Cohn-Kanade (CK+) el cual contiene 593 imágenes de expresiones posadas y no posadas de 123 sujetos y expresiones faciales femeninas japonesas (JAFFE) que conta de 213 imágenes de 10 modelos femeninas japonesas. Se seleccionaron al azar 80% de las muestras para el entrenamiento y el 20% restante se utilizaron de prueba. Se realizaron oclusiones en el 50% de las muestras del conjunto entrenamiento y de igual forma ocurrió con el conjunto de pruebas.

Dentro de los experimentos que se realizaron se encuentran los siguientes métodos: PCA + K-NN, PCA + LDA + K-NN, PCA + SVM y PCA + LDA + SVM; obteniendo los resultados como se puede apreciar en la tabla 2.1 cuyos valores refieren al promedio de la precisión en el reconocimiento.

Tabla 2.1 Precisión promedio con respecto al método empleado (Cornejo & Pedrini, 2016).

	Non-occluded facial images Accuracy %					**Occluded facial images Accuracy %**			
	Method	**LBP**	**CENTRIS**	**LGC**	**LGC-HD**	**LBP**	**CENTRIS**	**LGC**	**LGC-HD**
CK+	PCA+K-NN	43.74	**55.82**	37.70	36.81	42.06	**54.04**	36.35	33.81
	PCA+LDA+KNN	92.62	**93.66**	83.44	87.17	88.06	**90.30**	78.06	80.18
	PCA+SVM	77.17	**81.27**	71.80	69.78	75.01	**78.51**	67.97	66.65
	PCA+LDA+SVM	92.84	**94.10**	83.89	85.82	88.44	**90.01**	78.51	79.33
JAFFE	PCA+K-NN	64.41	**87.50**	66.91	73.10	45.84	**87.50**	66.91	74.06
	PCA+LDA+KNN	**93.00**	91.60	83.10	88.34	83.10	**91.60**	83.10	88.57
	PCA+SVM	84.18	83.40	85.84	**86.31**	70.60	83.40	85.84	**88.67**
	PCA+LDA+SVM	**92.50**	92.00	82.03	87.51	81.44	**92.00**	82.03	87.47

De acuerdo con los experimentos, se observa que el operador CENTRIS logra más del 90%, observándose que el *dataset* CK+ CENTRIS siempre tiene un desempeño superior a los otros métodos de extracción de características. En el caso de JAFFE, LBP es ligeramente superior a CENTRIS para imágenes no ocluidas; sin embargo, el operador CENTRIS es mucho mejor entre los operadores de textura para imágenes ocluidas. Se observa también que LGC-HD proporciona mejores resultados que el operador LGC y que combinar las técnicas PCA+LDA proporciona una mejor precisión que solo aplicar PCA.

Finalmente, compararon su enfoque propuesto (CENTRIS+PCA+LDA+SVM) con otros descriptores y logra mejores resultados para los *datasets* CK+ y JAFFE bajo oclusión. ver tabla 2.2.

Tabla 2.2 Comparación con otros sistemas (Cornejo & Pedrini, 2016).

	Accuracy %			
	Approach	**Strategy**	**Non-oc**	**Oc**
CK+	Proposed method	CENTRIS+PCA+LDA+SVM	94.10	90.01
	Proposed method	LBP+PCA+LDA+SVM	92.62	88.44
	Ramirez et al. [27]	Gabor+PCA+LDA+SVM	94.03	85.68
	Zhang et al. [32]	Maximum Likelihood Estimation	94.29	85.24
		Sparse Representation		
JAFFE	Proposed method	CENTRIS+PCA+LDA+SVM	92.00	92.00
	Liu et al. [31]	Maximum Likelihood Estimation	93.42	86.73
		Sparse Representation		
	Proposed method	LBP+PCA+LDA+SVM	93.00	83.10
	Ramirez et al. [27]	Gabor+PCA+LDA+SVM	95.12	82.86
	Zhang et al. [32]	Gabor template and SVM	81.20	48.80

Validación Experimental de la Influencia de Oclusiones en Reconocimiento Facial (Gonzalez-Sosa et al., 2016)

Se desarrolló una propuesta para trabajar la oclusión facial mediante la fusión de regiones del rostro no obstruido haciendo uso de Face++[5] y del sistema de patrones binarios locales (LBP) (Ahonen et al., 2006) considerando diversas escalas y trabajando con la *dataset* ARFace (Aleix Martinez & Benavente, 1998) ya que contiene oclusiones reales para imágenes en 2D y se consideran tres escenarios: neutral, gafas de sol y bufandas.

ARFace contiene imágenes de 136 sujetos de los cuales 76 son hombres y 60 mujeres con un total de 26 imágenes para cada sujeto. Dichas fotografías presentan variaciones con respecto a las expresiones, iluminación y oclusiones, integrando un total de 3,300 imágenes de *576x768* píxeles.

Se utilizó Face++ con el fin de hacer uso del módulo de detección (API para comparar las caras) con las localizadas individualmente, el cual entrega una puntuación de similitud y es capaz de proporcionar cuatro regiones faciales: cejas, ojos, nariz y boca.

Se dividió la región facial en bloques de *10x10* y calculando el histograma para cada uno, para, posteriormente, ser trabajados con escalas de *2, 1, 0.5, 0.25* y *0.125*; los

5 Face++ - Face++ Cognitive Services (faceplusplus.com)

concatenaron y aplican la técnica de Análisis de componentes principales para reducir sus dimensiones

Posteriormente, se dividió el *dataset* en dos grupos distintos, tabla 2.3. Evalúan los tres escenarios diferentes: neutral, gafas de sol y con bufanda utilizando imágenes con iluminación controlada.

Tabla 2.3 Conjuntos de Evaluación y desarrollo (Gonzalez-Sosa et al., 2016).

Conjunto	Identidades Hombre	Identidades Mujeres
Desarrollo	1-49	1-39
Evaluación	50-76	40-60

Analizando los resultados mencionan que la fusión de todas las regiones faciales no siempre es la mejor opción, ya que el rendimiento tiende a deteriorarse. Considerando la región de los ojos y la nariz como buenas; por ejemplo, para la nariz, aumenta 2.56% de EER a 8.12% de EER con Face++, mientras que el 12.82% de EER aumenta a 27.67% con LBP. También se observó que Face++ tiene un mejor desempeño que LBP en todos los escenarios, ver tabla 2.4.

Tabla 2.4 Resultados obtenidos (Gonzalez-Sosa et al., 2016).

		Individual Regions				Fusion	
Sistema	Escenario	Cejas	Ojos	Nariz	Boca	Cuatro Regiones Faciales	Regiones Faciales No Oclusas
Face++	Neutral	7.69	7.69	2.56	7.69	2.56	**2.56**
Face++	Gafas de sol	50.71	46.42	33.57*	**12.14***	20	17.14
Face++	Bufanda	15.97*	17.85*	**8.12***	54.46	11.6	11.56
MultiscaleLBP	Neutral	15.38	7.69	12.82	10.25	4.58	**4.58**
MultiscaleLBP	Gafas de sol	41.46	45.71	22.14*	14.28*	12.85	**11.80**
MultiscaleLBP	Bufanda	16.07*	12.5*	27.67*	49.95	10.71	**10.71**

Fast and Robust Occluded Face Detection in ATM Surveillance (T. Zhang et al., 2018)

Los autores se centraron en el problema de la oclusión severa en la vigilancia en cajeros automáticos (ATM), proponiendo un algoritmo para detectar la oclusión del rostro implementando la silueta Omega que se forma entre la cabeza y los hombros de las personas. Los autores dividen en 3 partes la contribución:

- Usan la forma Omega formada por la cabeza y los hombros de las personas.
- Desarrollo del algoritmo para el seguimiento de la cabeza usando señales de gradiente.
- La combinación de un clasificador de color de piel y el clasificador de coincidencia con el algoritmo AdaBoost.

Observaron que la forma Omega formada entre la cabeza y los hombros se parecía a una distribución gaussiana, diseñaron un algoritmo para detectar la cabeza que se basa en la energía potencial y consta de 3 pasos principales: a) Inicialización de la posición, b) Proceso iterativo basado en minimizar una energía potencial y c) Diseño de condiciones de restricción. Posteriormente, aplican el operador Canny para obtener los bordes y ajustarlos en una elipse con los contornos de los límites de la cabeza, ver figura 2.3.

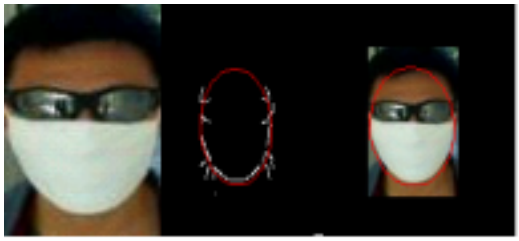


Figura 2.3 Ejemplo de ajuste de elipse (T. Zhang et al., 2018).

La evaluación del algoritmo propuesto se llevó a cabo en 120 secuencias de video de 8 sujetos imitando los videos capturados por los cajeros automáticos con cabezas ocluidas y bajo un fondo complejo. De entre los 120 videos se pueden encontrar casos sin y con oclusión por un sombrero, oclusión bocal por máscara y oclusión ocular por lentes. Finalmente, reportan los siguientes resultados con respecto a la detección: 97.95% para rostro normales, 93.57% para rostros ocluidos y un 94.35% general. 96.70% usando el color de la cara, 94.35% usando la plantilla de la cara y aumenta hasta un 98.56% si se implementa un clasificador en cascada.

Pixel-Level Occlusion Detection Based on Sparse Representation for Face Recognition (Zhao, 2018)

En el artículo se presenta un sistema para reconocer personas que considera la detección de oclusiones a nivel píxel. Los autores utilizan el método denominado PLOD_FR que se basa en una representación dispersa (SRC) de una imagen de consulta y estimar si hay presencia de oclusión o no en cada píxel, para obtener una estimación integrada.

Mencionan que para que la detección de la oclusión sea precisa y poder eliminar coeficientes negativos se suma el valor del residuo final con el valor absoluto mínimo de todos los residuos. Para cada clase de residuos se establece un umbral *th1*, donde si un píxel que está siendo evaluado es oscuro tendrá un valor residual mayor al umbral t*h1*; de lo contrario se considera píxel sin oclusión. Para regularizar la región ocluida se aplica una operación morfológica de dilatación y finalmente la

clasificación se realiza con respecto a la clase que tiene el residuo más bajo. La propuesta se comparó con otros 5 métodos.

Para la experimentación se trabajó con los *datasets* AR, Yale B y PIE: AR consta de 3,000 imágenes de 120 personas (65 hombres y 55 mujeres) y considera imágenes con gafas y bufandas. Mencionan que para imágenes con gafas de cada persona seleccionan diferentes números de muestras para el entrenamiento, el número de muestras crece a medida que el reconocimiento mejora. Los parámetros utilizados son: lambda con valor de *0.1*, $th1 = 4 * \frac{mr}{3}$ y $th2 = 0.98 * C$ y $disk', 1$ como elemento estructurante para la dilatación. Los resultados obtenidos cuando el rostro está ocluido con gafas de sol se observan en la tabla 2.5, y cuando las personas utilizan bufanda se muestran en la tabla 2.6.

Tabla 2.5 Comparativa de oclusión con lentes de sol (Zhao, 2018).

Dimension Methods	**180**	**320**	**500**	**720**
PLOD_SR	0.9861	0.9945	0.9972	0.9903
SRC	0.9542	0.9695	0.9750	0.9819
WGSRC	0.9139	0.9764	0.9806	0.9848
CRC	0.8875	0.9431	0.9486	0.9722
NFS	0.9472	0.9667	0.9764	0.9833
NN	0.9384	0.9556	0.9611	0.9653

En la tabla 2.5 se presenta una comparativa entre cinco métodos, además del método presentado por los autores para trabajar la oclusión entorno a utilizar lentes de sol, en este caso se observa que el método propuesto es el mejor en tasas de reconocimiento facial y también se muestra robustes antes diversas dimensiones

Tabla 2.6 Comparativa de oclusión con bufanda (Zhao, 2018).

Dimension Methods	**180**	**320**	**500**	**720**
PLOD_SR	0.7218	0.7375	0.7681	0.7666
SRC	0.2791	0.3320	0.3611	0.3667
WGSRC	0.7778	0.7833	0.8083	0.7981
CRC	0.6167	0.6764	0.6930	0.6861
NFS	0.2417	0.2820	0.3111	0.2986
NN	0.2694	0.3042	0.3375	0.3278

En la tabla 2.6 muestra las tasas obtenidas de la comparativa de los 6 métodos para el reconocimiento facial bajo la oclusión de bufanda, se observa que el método propuesto por los autores y WGSRC detectan mejor la oclusión, logrando que se realice el reconocimiento facial correctamente hasta en un 76.81% por el método propuesto y hasta 80% por el método WGSRC con una dimensión de 500 en ambos casos, siendo ligeramente más bajas las tasas del método propuesto que WGSRC.

Los autores mencionan que su método junto con WGSRC, logran la tasa de reconocimiento más alta ya que intentan detectar y disminuir el efecto causado por la oclusión; sin embargo, mencionan que el método debe mejorar el proceso de dividir la imagen en bloques, ya que esto puede ahorrar mucho tiempo y ser práctico en la aplicación real.

Con respecto al dataset Yale B, se integra de 38 personas en 9 poses y 64 condiciones de iluminación. Eligieron aleatoriamente 7 imágenes como muestra de entrenamiento y 3 con oclusión como prueba. Este proceso lo repitieron 5 vece usando los valores medios como tasas de reconocimiento finales, usando lambda con valor de *0.01*, $th1 = 4 * \frac{mr}{3}$ y $th2 = 0.98 * C$ y $streal\ disk', 1$ como elemento estructurante para la dilatación. Mencionan que su método es mejor que WGSR.

Tabla 2.7 Resultado de oclusión del rostro lado derecho (Zhao, 2018).

Methods	**PLOD_FR**	**WGSR**	**CRC**	**SRC**	**NFS**	**NN**
Results	0.9895	0.9105	0.9795	0.9632	0.9789	0.5368

Tabla 2.8 Resultado de oclusión del rostro lado izquierdo (Zhao, 2018).

Methods	**PLOD_FR**	**WGSR**	**CRC**	**SRC**	**NFS**	**NN**
Results	0.9263	0.8074	0.9	0.6421	0.8211	0.2684

En la tabla 2.7 se puede apreciar el resultado del método al aplicar oclusión en el lado derecho causada por la iluminación y en la tabla 2.8 el resultado del lado izquierdo del rostro igualmente causada por la iluminación, en ambos casos el método propuesto resulta ser más robusto en el reconocimiento facial antes oclusión por cambios de iluminación obteniendo hasta un 98% en el lado derecho y un 92% en el lado izquierdo del rostro, también se observa que algunos métodos obtienen tasas bajas de reconocimiento dado a las características de detección de cada uno ante la oclusión de iluminación.

Dataset PIE: se forma de 68 personas, con de 5 poses y con cambios de iluminación. Se eligieron imágenes de caras frontales sin oclusión para entrenamiento y para la evaluación consideraron aquellas con presencia de oclusión. Pero dado que son imágenes no alineadas, no funcionan correctamente.

Sin embargo, mencionan que su método de reconocimiento logra tasas más altas que otros métodos, en la tabla 2.9 se pueden apreciar los resultados de reconocimiento aun ante cambios de pose, observándose que en el método propuesto logra tasas de reconocimiento altas hasta en un 66% con imágenes que no se encuentran bien alineadas lo que causa bajo rendimiento en los demás métodos.

Tabla 2.9 Resultados de oclusiones por pose (Zhao, 2018).

Dimension	169	676	1521
PLOD_SR	0.4639	0.6489	0.6698
SRC	0.4960	0.5628	0.5668
CRC	0.5468	0.4733	0.4238
NFS	0.2901	0.3262	0.3195
NN	0.1952	0.2219	0.2193
SPP	0.1511	0.1457	0.1025

2.2.2 Detección del Rostro

Face Recognition with Occlusion Using a Wireframe Model and Support Vector Machine (Garcia Rios et al., 2017)

Se propuso un sistema para el reconocimiento facial implementando un modelo wireframe y la herramienta Facefit (Kawamoto et al., 2004) el cual tiene una estructura geométrica *3D* del rostro para extraer un sistema de coordenadas que integran la información relevante del rostro para posteriormente, ser examinada en segmentos y dividirlos en grupos con el fin de localizar cada una de las partes principales del rostro como es la nariz, boca, ojos y cejas y poder omitir datos las zonas que presenten algún tipo de oclusión.

El sistema propuesto se basa en dos procesos: uno para el entrenamiento de los datos para identificar y verificar a la persona que ingresa al sistema. Las imágenes de entrada son sometidas a un recorte para eliminar el fondo y dejar la parte que contiene al rostro y pasarla a un mapa de bits normalizado a *520x480* píxeles.

Se crea un modelo de malla con 174 puntos y 759 coordenadas por eje, ver figura 2.4, empleado el Modelo de Apariencia Activa el cual consiste en un conjunto de coordenadas de superficies triangulares. La apariencia se define como la intensidad de cada uno de los píxeles dentro de la malla a los cuales se les obtienen vectores de forma y se le aplica el análisis de componentes principales para obtener un vector promedio también llamado vector de valores de deformación.

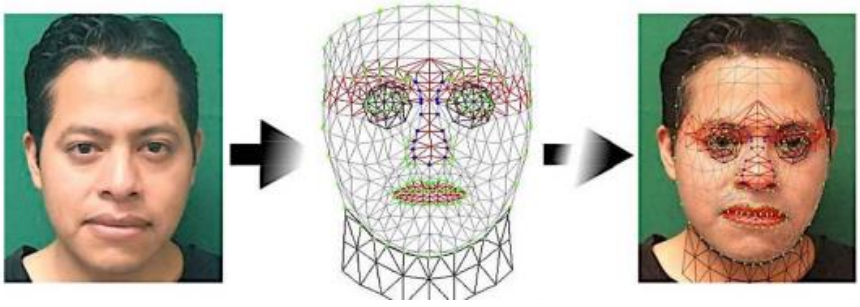


Figura 2.4 Creación del Modelo de Malla del rostro (Garcia Rios et al., 2017).

La oclusión pude dar datos erróneos desproporcionando el rostro. Una vez que se tienen los puntos característicos de la imagen del rostro, con o sin oclusión, se procesan los datos en el proceso de aprendizaje empleando máquinas de soporte vectorial (Smola & Schölkopf, 2004) y posteriormente se pasan a LIBSVM library (Chang, 2011) para la identificación y verificación, creando un número de modelos acorde al número total de personas en el entrenamiento del sistema.

El *dataset* empleado contiene 900 imágenes estéreo de un total de 60 personas con 15 imágenes por persona y, para cada caso a tratar de oclusión en los ojos, boca, parte derecha e izquierda del rostro se entrenó el clasificador SVM. El tener imágenes estéreo permitió tener resultados que marcan diferencia ya que Facefit no es capaz de considerar las zonas de los oídos y al tener una imagen estéreo, la propuesta entrega un comportamiento favorable dando valores arriba del 95%.

Se comparó el sistema con Eigenphases (Olivares-Mercado et al., 2016) el cual obtiene un 80.63% en la identificación, considerando únicamente los cambios de iluminación y un 96.28% en la identificación con una variación de oclusiones. También se comparó con un método de reconocimiento de rostros invariante a iluminación, mediante la aplicación de filtros recupera información oculta por los cambios de iluminación (Nabatchian et al., 2011).

Tabla 2.10 Resultados obtenidos de los tres métodos de reconocimiento analizados (Garcia Rios et al., 2017).

Tipo de oclusión	Método 1 [13]	Método 2 [15]	Nuestro
Gafas obscuras	91.42%	94.50%	96%
Bufanda	93.49%	88.67%	95%

Los resultados obtenidos con dos tipos de oclusión (gafas obscuras y bufanda) se pueden observar en la tabla 2.10 en comparativa con dos métodos de reconocimiento, analizando los dos tipos de colusión antes mencionados obteniendo un promedio de los porcentajes, se logra observar que con el método propuesto se obtiene un porcentaje promedio mayor de hasta el 96% con gafas obscuras y 95% con bufanda, se menciona que es estable al omitir parte de los valores obtenidos por el modelo sin perder la caracterización del rostro en el reconocimiento.

Techniques and Challenges of Face Recognition: A Critical Review (Singh & Prasad, 2018)

Se realizó una recopilación de algunas técnicas empleadas en el reconocimiento facial implementadas en imágenes *2D* y *3D*, considerando varios desafíos como es

el factor de envejecimiento, rasgos faciales, variaciones de pose, la oclusión y la iluminación.

Revisaron los *dataset* ORL, YALE, PHPID, VLC, FRGC y LFW. En dicho estudio se encontró que con LFW existe una tasa de precisión máxima de 99.33% y con el *dataset* de rostros UGC-JU se alcanza un 99.07% mediante clasificadores Random Forest.

Los autores mencionan mejorar la tasa de reconocimiento facial al agregar información de profundidad. Y para trabajar la oclusión en grandes *datasets* se basaron en la oclusión genérica y el enfoque de diccionario disperso estructurado, observando que la tasa de rendimiento y reconocimiento en imágenes *2D* y *3D* son casi iguales; pero, para trabajar con *3D* hace uso de la GPU.

Los resultados de algunas de las técnicas de reconocimiento facial se pueden observar en la tabla 2.11.

Tabla 2.11 Comparación de diferentes técnicas de reconocimiento facial, aplicaciones y tasa de precisión (Singh & Prasad, 2018).

No.	Techniques	Applications	Face database applied	Accuracy/Recognition rate
1	Histogram Oriented Gradient (HOG) [9]	Global gabor/Zernike descriptor	ORL YALE AR	98% 97.80% 97.10%
2	Featural Processing [10]	2D/3D Face Recognition	Stereoscopic information	Higher in 3D Face
3	Fusion algorithm RF classifier [19]	Visible and Thermal Imagen Faces	UGC-JU	99.07%
4	Two-dimensional multicolour fusion [22]	2D-MCF Model/Partitioned sparce sensing recognition (P-SRC)	AR CURTIN FRGC Bosphorus	Up to 24.5% improved Up to 3.8% improved Up to 25% improved Up to 2.86% improved
5	Modified Local Directional Patterns [27]	General Discriminant Analysis	Deep Belief Network	96.25%
6	Multi-scale strategy based on geometric and local descriptor [36]	3D face recognition	GavabDB and Bosphourus	98.90%
7	LBP technique, shape model [33]	3D face recognition	PHPID database VLC database	88.76% 44.97%
8	Local geometric feature and shape matching [32]	3D face recognition	FRGCv2 database	97.00%
9	Face descriptor based on Gabor filter [37]	Face recognition in wild animal	LFW dataset	97.29%
10	Domain adaptation [38]	Face recognition in wild animal	LFW dataset	99.33%

Occlusion Robust Face Recognition Based on Mask Learning With Pairwise Differential Siamese Network (Song et al., 2019)

Las redes neuronales convolucionales son clave para el reconocimiento facial, pero presentan dificultades ante variaciones como la posición de captura, la expresión facial, la iluminación y la oclusión. Como medida de solución los autores proponen una estrategia de aprendizaje para el reconocimiento facial y poder descartar elementos corruptos.

Establecen un diccionario de máscaras con las principales diferencias en las características de caras ocluidas y libres de oclusión haciendo uso de una red siamesa diferencial por pares (PDSN) entrenadas mediante binarización. Cada elemento del diccionario es la captura de la correspondencia entre las áreas faciales ocluidas y los elementos característicos dañados, denominado máscara de descarte de características (FDM). Los autores proponen aprender de un generador de máscaras minimizado una combinación de dos pérdidas:

- Una pérdida de contraste por pares, penalizando las grandes diferencias entre las características de caras limpias y ocluidas.
- Una pérdida de clasificación que asegura que aquellos elementos que dañan el reconocimiento estén enmascarados.

El modelo descompone el reconocimiento facial con oclusiones en 3 etapas:

- *Etapa I*: Generadores de máscaras de aprendizaje utilizando la red siamesa diferencial por pares propuesta (PDSN)

- *Etapa II*: Establece un diccionario de máscaras a partir de PDSN.

- *Etapa III*: combinación de la máscara con descarte de características (FDM) de oclusiones parciales aleatorias del diccionario.

Algunos de los objetos que ocluyen y que forman parte del *dataset* FacesCrub son las gafas de sol, una máscara, la mano, un antifaz, una bufanda, un libro, un teléfono, una taza, un sombrero, etc. En la figura 2.5 se puede observar una descripción general de la propuesta del artículo.

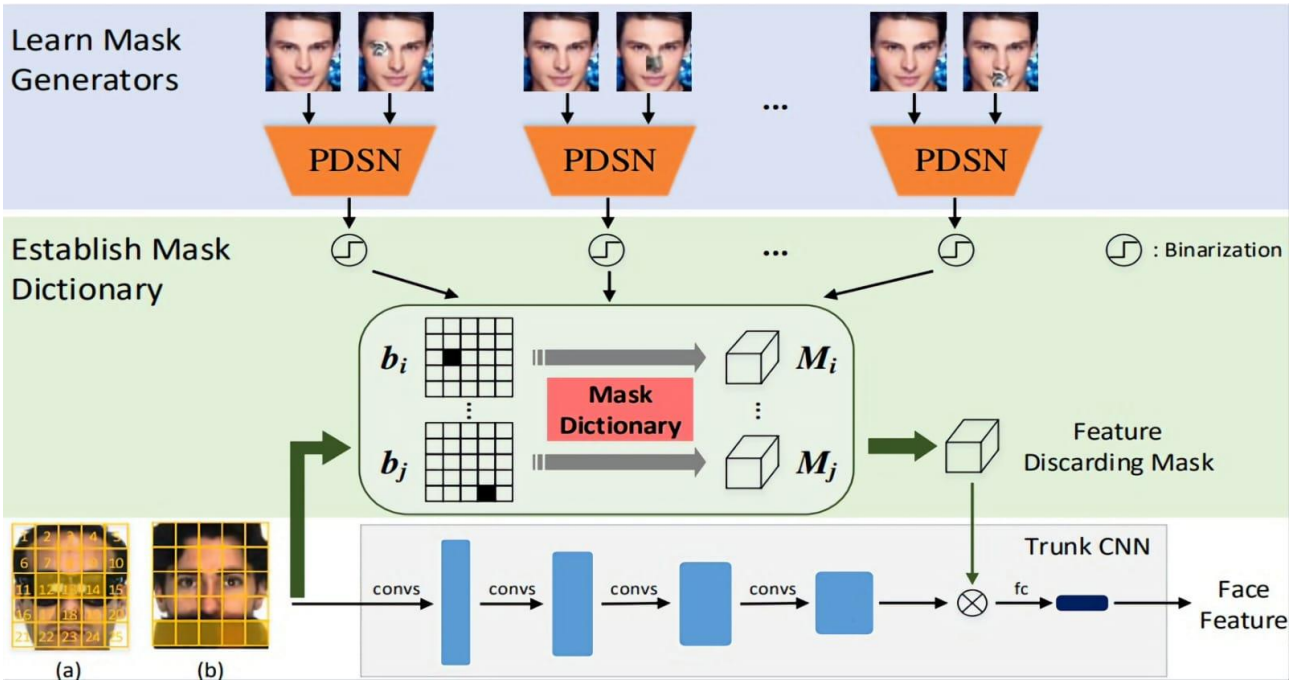


Figura 2.5 Una descripción general del marco propuesto basado en un modelo de CNN troncal entrenado para el reconocimiento facial, proponen la estructura de red siamesa diferencial por pares (PDSN) para aprender la correspondencia entre bloques faciales ocluidos y elementos característicos corruptos. Luego, se establece un diccionario de máscaras en consecuencia, que se utiliza para integrar la máscara de descarte de características (FDM) para un rostro de prueba con oclusiones parciales aleatorias. Finalmente, multiplican el FDM con características faciales originales para eliminar los elementos de características dañados del reconocimiento (Song et al., 2019).

Facial Detection and Recognition System Using Hybrid Techniques in Images and Video Sequences (Cutipa et al., 2019)

Proponen la detección y el reconocimiento automático del rostro mediante la extracción de características de los Filtros Gabor y su combinación con los componentes principales. El clasificador en cascada basado características Haar es efectivo para la detección de objetos y utilizan el clasificador máquinas de soporte vectorial.

Contemplan cuatro etapas: la adquisición de la imagen, la detección de rostros, extracción de características y la clasificación. La captura de la imagen de realiza en el espectro de colores RGB y se reducen a *240x320* píxeles en escala de grises. Par el clasificador Haar utilizan un kernel de *20x20*, y aplican 40 filtros Gabor a la imagen, en ocho escalas, integrando el vector de características.

El sistema logró un 98% en la detección y el reconocimiento de rostros superando el 95% esperado y obteniendo un 2% de falsos aceptados y falsos rechazados, mediante la combinación de técnicas híbridas. Utilizó los *datasets* FERET, ORL y uno propio.

Masked Face Recognition Dataset and Application (Wang et al., 2020)

Derivado de la pandemia mundial provocada por el COVID-19 y el uso de mascarillas para la prevención del contagio de este, los sistemas de seguridad basados en el reconocimiento facial convencional son ineficaces en muchos casos. Los autores presentan el desarrollo de un modelo para el reconocimiento de personas con imágenes de rostros enmascarado que se basa en el rostro y en los ojos con una precisión de reconocimiento del 95%.

Los autores mencionan que, dependiendo de los escenarios aplicativos, se encuentran algunas dificultades, ejemplo:

- Entornos de aplicación no controlados: videovigilancia pública en donde la distancia de captura, la pose, la oclusión y la iluminación son inciertas y muy variantes.

- Entornos de aplicación controlados: Existen lugares donde todo es controlado, por ejemplo, lugares de trabajo, controles de seguridad en aeropuertos y estaciones del tren y con ello se tienen ciertas condiciones en la adquisición de las imágenes tales como cooperación de los usuarios con fotografías frontales y cámaras de alta resolución.

Se utilizaron tres tipos de *datasets* de rostros enmascarados:

1. Detección de rostro enmascarado *dataset* (MFDD). Contiene 24,771 imágenes de rostros enmascarados.
2. Reconocimiento facial enmascarado del mundo real *dataset* (RMFRD). Incluye 5,000 fotografías de 525 personas con máscaras y 90,000 imágenes de los mismos 525 sujetos sin máscaras.
3. Reconocimiento de rostro enmascarado simulado *dataset* (SMFRD).

Para mejorar la eficiencia del sistema, desarrollaron un software de uso de mascarillas, basado en la biblioteca Dlib, para simular el uso de máscaras automáticamente. De esta manera, también contribuyeron creando un *dataset* de caras enmascaradas simuladas, que se integra de 500,000 imágenes de rostros de 10,000 personas que se encuentran en el *dataset* (SMFRD) ver figura 2.6.

Figura 2.6 Ejemplos de un par de imágenes faciales (a) y (b) son imágenes faciales normales. (c) y (d) son imágenes de rostros enmascarados de manera correcta e incorrectamente, (Wang et al., 2020).

Su propuesta consiste en el reconocimiento de multi-granularidad basado en rostro-ojo; es decir, extraen características de las partes visibles del rostro enmascarado tales como el contorno del rostro, el ojo, área peri-ocular (zona alrededor del ojo como las bolsas debajo de los ojos) frente y les dan un peso dependiendo de su información discriminativa.

Los autores concluyen que el algoritmo tiene aplicaciones de reconocimiento facial sin contacto, en escenarios que requieren de una validación de autentificación. Además, debido a la contaminación y enfermedades como la gripa, las personas usan mascarilla y la necesidad de reconocimiento facial con mascarilla persistirá durante mucho tiempo.

Masked Face Recognition Using Deep Metric Learning and Facemasknet-21 (Golwalkar & Mehendale, 2020)

Los autores presentan una red llamada FaceMaskedNet-21 para el desarrollo del sistema. Utilizan bibliotecas estándar como openCV, python y deep learning. El sistema está enfocado a imágenes estáticas, así como archivos de video ya que funciona con un tiempo de ejecución aproximadamente de 9.52 ms.

La técnica de aprendizaje profundo se utilizó para proporcionar un vector de características de salida de *128 dimensiones*, que fue logrado por una red FaceMaskNet-21 entrenada usando un conjunto de imágenes llamado cuatrillizo ver figura 2.7.

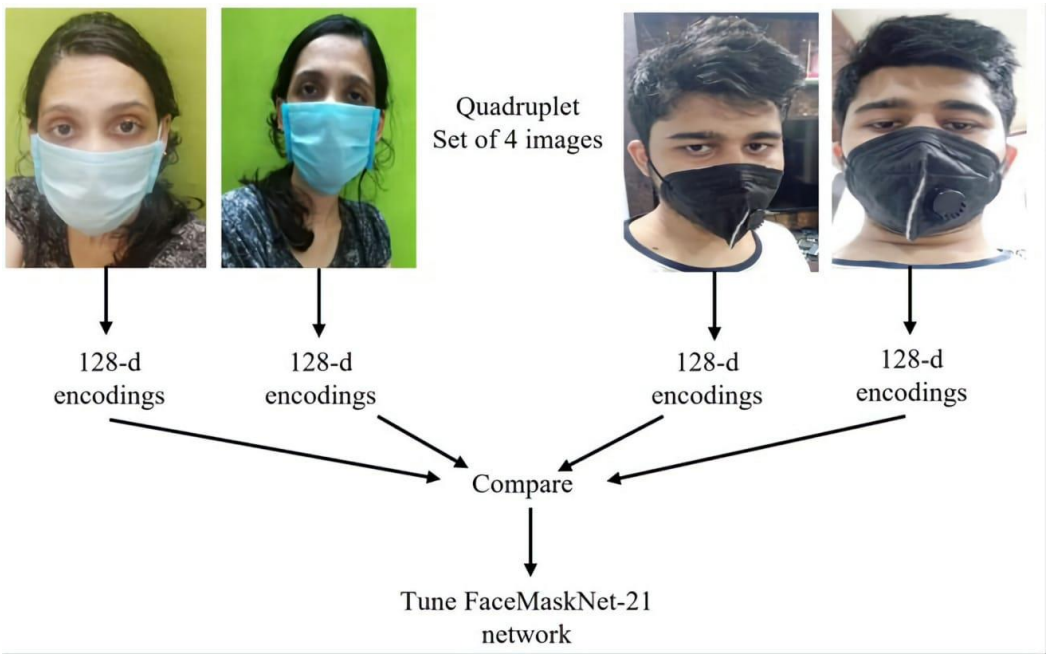


Figura 2.7 Proceso de ajuste de la red FaceMaskNet-21 cada imagen en el cuatrillizo se convierte en codificaciones de 128-dimensiones y estas codificaciones faciales se comparan entre sí para sintonizar la red FaceMaskNet-21 (Golwalkar & Mehendale, 2020).

Inicialmente la arquitectura de red FaceMaskNet-21 se entrenó con aproximadamente 50,000 imágenes utilizando el *dataset* de caras etiquetadas en la naturaleza (LFW). Para posteriormente, optimizarla con un *dataset* de rostros enmascarados logrando reconocer correctamente incluso si tiene cubierta la mitad de la cara, ver figura 2.8. Emplean la técnica HOG para el reconocimiento facial seguida de la distancia euclidiana para una comparación de los codificadores de caras de entrada con respecto a los almacenados en el *dataset.* Tienen una precisión general del 88,92%.

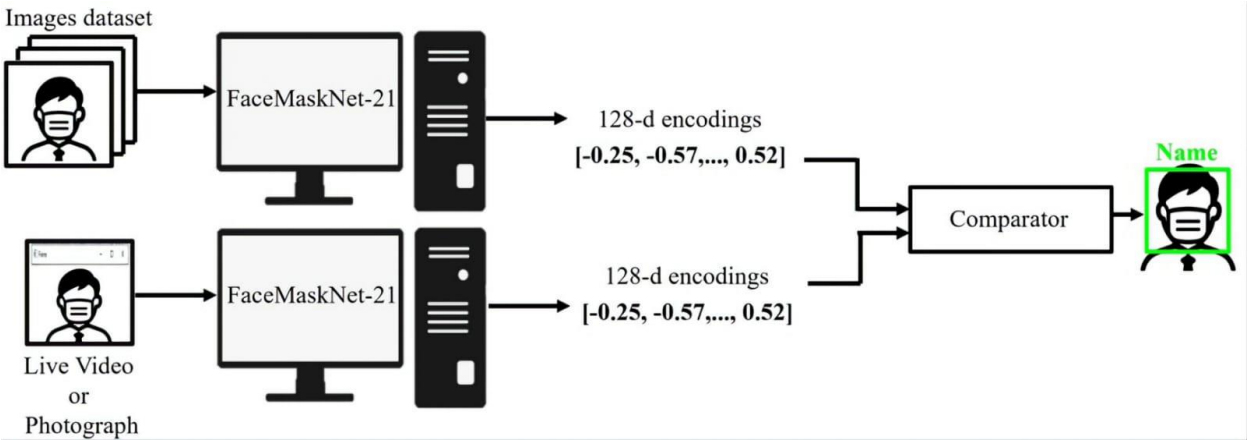


Figura 2.8 Diagrama conceptual para el reconocimiento facial enmascarado FaceMaskNet-21 convierte imágenes del conjunto de datos y la imagen de entrada o video en vivo en codificaciones de 128 dimensiones. Se comparan las dos codificaciones y se muestra el nombre reconocido (Golwalkar & Mehendale, 2020).

En la figura 2.9 se puede apreciar que una imagen de entrada RGB de tamaño *227X227*, se pasa a través de la capa de normalización del canal cruzado convolucional + ReLU + de dimensiones *55X55* y que consta de 96 filtros. Después de esto, se pasa por la capa de agrupación convolucional + ReLu + Max de dimensiones *27X27* y que consta de 96 filtros. Posteriormente, ingresa a través de la siguiente capa de agrupación convolucional + ReLu + Max de dimensiones *13X13* y que consta de 384 filtros. Finalmente, entra a través de 100 capas completamente conectadas seguidas de una capa Softmax que produce *128 codificaciones* de salida.

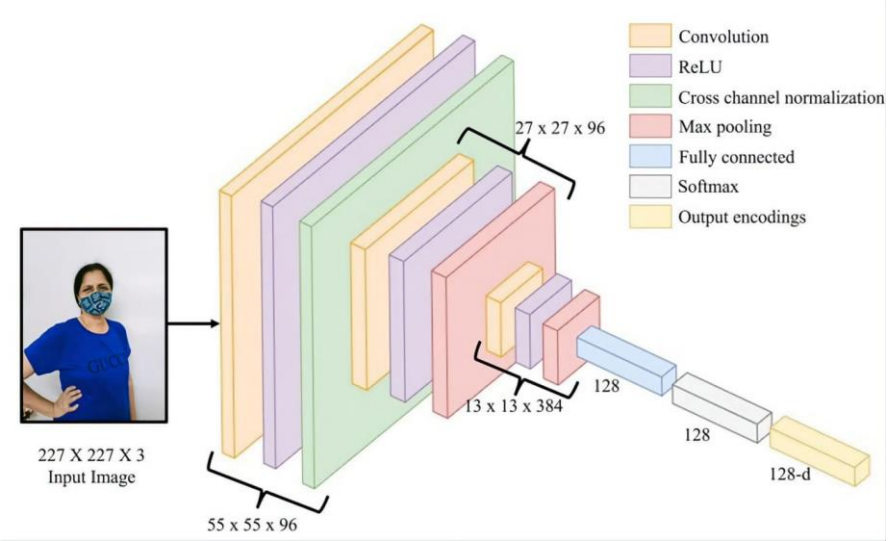


Figura 2.9 Arquitectura de red FaceMaskNet-21 (Golwalkar & Mehendale, 2020).

Se probó con 325 imágenes y 7 videos. El sistema identifica al 100% con una sola cara, pero falla al intentar detectar localizar los rostros cuando hay más de una persona con mascarilla presentés en la escena.

¿Llevas Puesta la Mascarilla? Un Software de Reconocimiento Está Listo para Chequear si las Personas Cumplen con el Correcto Uso (Wudan, 2020)

Derivado de la pandemia se busca implementar sistemas que verifiquen el cumplimiento de las medidas sanitarias en espacios públicos y privado. Como medida de prevención de contagio, empresas como Apple lanzó la actualización de su sistema de reconocimiento facial que al detectar una mascarilla en el rostro solicita al usuario ingrese su contraseña para evitar que se quiten el cubrebocas y corran el riesgo de contagiarse.

En restaurantes y hoteles de Estados Unidos y Europa se ha implementado el algoritmo LeewatHertz para asegurarse de que el personal cumpla con la norma de

portar el cubrebocas. Sin embargo, la falta de normativas que regulen la información que pudiera ser obtenida preocupa y confunde a la ciudadanía lo que conlleva a aprobar leyes como es el caso de Portland, Oregón en donde se prohibido el uso público y privado del reconocimiento facial para proteger la privacidad de la población.

Towards Facial Recognition Problem in COVID-19 Pandemic (Mundial et al., 2020)

En el artículo se presenta una metodología de aprendizaje automático para el reconocimiento de rostros enmascarados de manera eficiente, mediante el clasificador Maquinas de Vector Soporte, logrando obtener un 97% de precisión en el reconocimiento de los rostros enmascarados.

También se implementó un método de selección de tripletes en línea durante la fase de formación y se presenta el *dataset* VGGFACE2[6] el cual contiene 9,500 clases diferentes y cada una de ellas contiene aproximadamente 330 imágenes procedentes de fuentes de internet y plataformas de redes sociales. Hacen uso de una PC con un I9 con GPU NVIDIA GeForce GTX 1660, cuyo entrenamiento con la biblioteca Caffe Deep Learning tuvo una duración de cinco días.

Inicialmente, con un conjunto de 800 imágenes (600 fotografías sin máscara y 200 con ella) de 200 personas se entrenó el modelo y se obtuvo un 79% el cual aumentó hasta 98% al incluir poses laterales de las personas enmascaradas y reentrenando el modelo. Sin embargo, si la imagen individual de la persona enmascarada presenta cierta rotación hacia la derecha o izquierda como se puede observar en la figura 2.10, el sistema es susceptible a generar resultados falsos. El modelo final mostró un 99% de precisión con el *dataset* VGGFACE2.

Figura 2.10 Pose irreconocible para clasificador enmascarado (Mundial et al., 2020).

6 VGGFACE2: https://zeus.robots.ox.ac.uk

Boosting Masked Face Recognition with Multi-Task ArcFace (Montero et al., 2021)

En este trabajo se desarrolló un enfoque basado en ArcFace usando LResNet50 como columna vertebral. Modifican la función de pérdida de ArcFace con la probabilidad de usar máscaras para obtener el Multi-Task ArcFace. Los autores mencionan que el modelo se entrenó usando dos GPU Tesla v100 con un lote de *512* y *3,000* pasos, utilizan el SGD como optimizador con un impulso de *0.9* y tasa de aprendizaje inicial de *0.0015* con una reducción de factor de *0.3* en pasos de *120K*, *200K* y *280K*.

Como *dataset* de entrenamiento se utilizó MS1MV2 el cual consta de 5.8 millones de imágenes de 85,000 sujetos y para la generación del *dataset* enmascarado se utilizó la herramienta MaskTheFace considerando los tipos de mascarilla quirúrgico, verde quirúrgico, azul quirúrgico, N95, tela y Kn95 de los cuales se seleccionan aleatoriamente con una probabilidad del 50% de aplicar un color aleatorio y 50% de colocar una textura aleatoria ver figura 2.11. También generan una nueva versión del *dataset* de gemelos enmascarados al realizar un aumento de datos.

Figura 2.11 Ejemplo de rostros enmascarados generados (Montero et al., 2021).

En la etapa de verificación, generaron versiones enmascaradas de 3 *datasets* conocidos de reconocimiento facial: LFW el cual contiene 13,233 imágenes de 5,749 personas y 6000 comparaciones para la verificación facial; CFP que contiene 500 celebridades de vista frontal (CFP FF) y perfil (CFP FP) que constan de 7,000 comparaciones respectivamente; y Agedb que se integra de 16,488 imágenes de 568 celebridades con un protocolo de 30 años con 6,000 comparaciones de los 4 diferentes protocolos que dispone este *dataset*.

Los resultados se pueden apreciar en la tabla 2.12, observándose que el método propuesto supera ampliamente ArcFace cuando se trata de rostros enmascarados. El rendimiento se hace más evidente con imágenes de perfil, también se observó la caída de rendimiento con respecto a ArcFace con caras no enmascaradas en menos del 2%. Finalmente, el modelo alcanza un 99.78% de precisión media lo que demuestra su eficiencia en esta tarea.

Tabla 2.12 Resultados del modelo propuesto (Montero et al., 2021).

Dataset	Proposed Method	Original Model
Masked LFW	98.92	94.75
Masked CFP_FF	98.33	92.73
Masked CFP_FP	88.43	76.81
Masked AGEDB_30	93.17	90.53
MFR2	99.41	97.17

En la tabla 2.12 se aprecia una comparativa del rendimiento entre el método propuesto por los autores y el modelo de ArcFace original con diversos *datasets*, logrando obtener hasta 99.41% de rendimiento con el método propuesto y el *dataset* MFR2 demostrando que el nuevo método es robusto ante la presencia de rostros enmascarados.

A Hybrid Deep Transfer Learning Model with Machine Learning Methods for Face Mask Detection in the Era of the COVID-19 Pandemic (Loey et al., 2021)

Los autores proponen un modelo híbrido que combina el aprendizaje automático clásico y profundo para la detección de máscaras faciales. El modelo propuesto se integra de cámaras de vigilancia para detectar a las personas que no llevan máscaras faciales e impedir la transmisión del virus. El sistema está conformado de dos componentes:

- Componente de extracción de características mediante Resnet50.
- Componente de clasificación de máscaras faciales con árboles de decisión, máquinas de vector soporte (SVM) y algoritmo de conjuntos.

Se utilizaron 3 *datasets* para el entrenamiento validación y pruebas.

- Rostros enmascarados del mundo real (RMFD) (Wang et al., 2020). Consta de 5,000 rostros enmascarados y 90,000 rostros desenmascarados.
- Rostros enmascarados simulados (SMFD) (prajnasb, 2020). Consta de 785 imágenes enmascaradas y 785 imágenes sin mascaras.
- Rostros etiquetados en la naturaleza (LFW) (Learned-Miller et al., 2016). Consta de 13,000 rostros enmascarados.

Determinaron que el clasificador SVM logró la mayor precisión posible en el menor tiempo arrojando los siguientes datos de precisión:

- En RMFD logró una precisión de prueba de 99.64%.
- En SMFD logró un 99.49% de precisión.
- En LFW logró 100%de precisión.

2.2.3 Aprendizaje Profundo

A Cascade Framework for Masked Face Detection (Bu et al., 2017)

Se propuso un sistema para la detección de rostros enmascarados, el cual consta de tres CNNs, además de un nuevo *dataset* denominado MASKED FACE para lograr un ajuste fino al modelo. El detector de rostros consiste en tres máscaras: *Máscara-12, Máscara-24-1* y *Máscara-24-2* las cuales son arquitecturas CNN binarias con la capacidad de clasificar de menor a mayor. *Mask-1* es una CNN de poca profundidad con 5 capas capaz de soportar imágenes de cualquier tamaño gracias a su estructura convolucional que evalúa la imagen de entrada con ventanas de detección de *12x12* con un espacio de 2 píxeles y asignará una probabilidad a la región analizada, si la probabilidad está por debajo del umbral será eliminada.

Posteriormente, se aplican *Mask-2* y *Mask-3* usando una ventana de detección de *24x24* y como ocurre en el caso de la primera máscara se calcula una probabilidad a las ventanas de detección que pertenezcan a una cara enmascarada. La probabilidad deberá ser mayor al umbral para permanecer si no, serán eliminadas y las que si superen el umbral serán las ventanas de detección final. Las probabilidades obtenidas por las *máscaras 2* y *3* deberán tener siempre un valor mayor de probabilidad.

Se utilizó el conjunto WIDER FACE (Yang et al., 2016) como *dataset* de preentrenamiento gracias a la gran cantidad de datos y su variabilidad en la escala, pose, oclusión e iluminación. Al ser modelos de clasificación binaria se necesita preparar dos categorías de entrenamiento el positivo y el negativo, para ello recortaron regiones rectangulares aleatoriamente cuya relación de inserción sobre unión (IoU) sea mayor a *0.65* para los positivos y con un valor inferior a *0.3* para los negativos, recolectando un total de 286,346 positivos y 859,038 negativos con una proporción de *1:3* para luego cambiar las imágenes a *12x12* para la *máscara 1* y a *24x24* para las dos restantes.

El *dataset* propuesto por los autores MASKED FACE, consta de 200 imágenes de internet de las cuales se seleccionaron 160 al azar para realizar el entrenamiento y las 40 imágenes restantes como el *dataset* de prueba. Para lograr un ajuste fino con estas imágenes, se recortaron regiones con relación de *0.65* IoU para los positivos y con valor de *0.*3 para los negativos recopilando 2,788 positivos y 8,364 negativos con proporción de *1:3*, para las *máscaras 2* y *3*.

Finalmente, se evalúa el método propuesto con el criterio de PASCAL VOC (Everingham et al., 2010) marcando como correcta una ventana final que sea mayor a *0.5* de IoU, logrando una precisión de 86.6% y recuperación del 87.7% como se puede apreciar en la figura 2.12.

Figura 2.12 Resultados de Masked Face detection algorithm (Bu et al., 2017).

Initial Shape Pool Construction for Facial Landmark Localization Under Occlusion (Wu et al., 2017)

Desarrollaron un sistema para localizar puntos de referencia bajo oclusión. Extraen *128* características invariantes de escala (SIFT) de cada rostro y aplican el algoritmo k-medias para obtener un conjunto de puntos o características fijas. Estos puntos son utilizados para generar la distribución de probabilidad del rostro, es decir, se puede tomar la probabilidad de un rostro latente. Y se aplica el muestreo de Gibbs para la clasificación.

Al comenzar el muestro de Gibbs, se le asigna una clase aleatoria a cada centroide, posteriormente, después de suficientes iteraciones de muestreo la distribución se mantendrá estable, es decir, converge. Teniendo las probabilidades de cada una de las clases se obtiene la pertenencia, donde un rostro se clasificará en la clase con la que tenga una mayor probabilidad.

En la experimentación se evaluaron las técnicas: RCPR, LBP-I-RCPR con el *dataset* COFW. Dicho conjunto cuenta con información sobre si el rostro está o no ocluido con un total de 29 puntos de referencias faciales. 1,345 imágenes se utilizaron para el entrenamiento y 507 para pruebas; mencionan que cada prueba se repitió por lo menos 10 veces.

Concluyen que el esquema propuesto se comporta mejor que LBP cuando $k<650$ y se observó que cuando $K=50$ se obtiene el error más pequeño, lo cual lograron reduciendo k de 25 a 55 con un intervalo de 10, indicando que no se reduce la precisión de la localización en los puntos de referencia del rostro, ver figura 2.13.

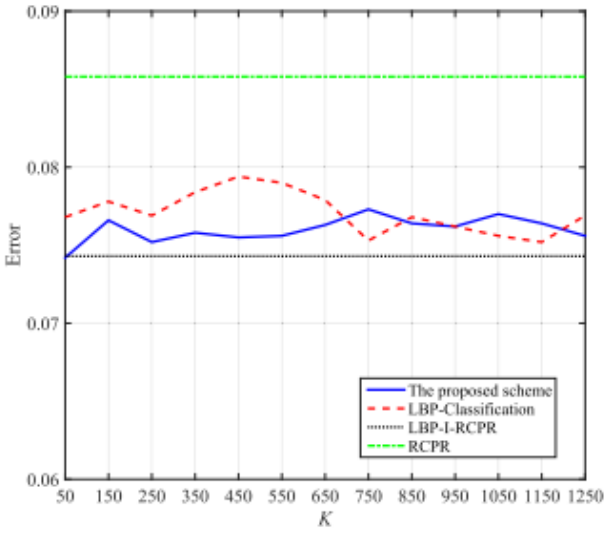


Figura 2.13 Comparación del error (Wu et al., 2017).

Se comparó el esquema propuesto con LBP-I-RCPR y RCPR con imágenes ocluidas obteniendo un 85.94%/40% de precisión/recuperación de LBP-I-RCPR, resultado más alto que RCPR con lo cual no se reduce la precisión en la detección de oclusión. Con respecto a la velocidad de respuesta, el esquema propuesto tiene 4.03 FPS cuando *k=50* pero no logra superar a RCPR ya que el esquema propuesto necesita analizar las correlaciones de los histogramas de LBP, siendo LBP-I-RCPR la forma más baja con 2.43 FPS. Mencionan que la velocidad se puede mejorar al implementarlo en C++ ya que todo lo desarrollaron en Matlab con un CPU de 2.20GHz limitando a si también la velocidad.

Patch-Gated CNN for Occlusion-aware Facial Expression Recognition (Y. Li et al., 2018)

Se propuso una red neuronal convolucional controlada por parches que localiza las regiones ocluidas en la cara de manera automática, creando un mapa de características de varios parches que se posicionan en puntos faciales de referencia. Además, entrenan la red para abordar en el reconocimiento de las expresiones del rostro las oclusiones, en la figura 2.14 se puede apreciar la idea principal de la propuesta.

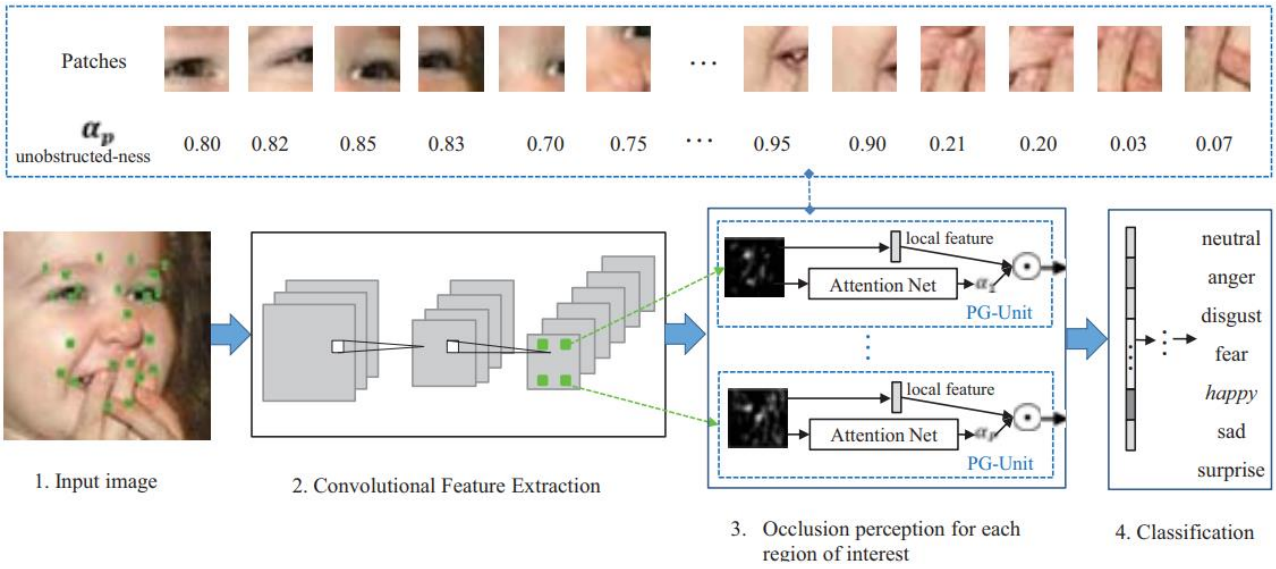


Figura 2.14 Estructura de PG-CNN (Y. Li et al., 2018).

El modelo aprende patrones de oclusión para codificarlos con pesos, permitiendo al modelo manejar cualquier objeto oclusor que se encuentre enfrente del rostro de la persona ya que descompone los mapas de características de toda la cara en 24 mapas de características para 24 parches locales, de un total de 68 puntos de referencia. Asigna pesos bajos para las partes ocluidas y pesos altos para las zonas que se encuentran libres y con ello con mayor discriminación.

La base para crear el modelo propuesto fue VGG 16 (Simonyan & Zisserman, 2014) por su desempeño en la clasificación de objetos y utilizaron para la evaluación los *datasets* RAF-DB (S. Li et al., 2017) y AffectNet (Mollahosseini et al., 2017) obteniendo los resultados que se muestran en la tabla 2.13.

Tabla 2.13 Datos obtenidos del test de precisión (Y. Li et al., 2018).

Accuracy % on RAF-DB and AFFECTNET		
Methods	RAF-DB (clean/occ.)	AffectNet (clean/occ.)
VGG-16	80.96/75.26	51.11/46.48
DLP-CNN	80.89/76.29	54.47/51.07
P-CNN	81.64/76.09	53.9/50.32
PG-CNN (proposed)	**83.27/78.05**	**55.33/52.47**

Occlusion Robust Face Recognition Based on Mask Learning (Wan & Chen, 2018)

Se desarrolló un modelo propuesto entrenable llamado Masknet el cual se puede incluir dentro de las arquitecturas de CNN existentes. Aprende de forma adecuada para la generación adaptativa de mapas características de imágenes faciales ocluidas, asignado pesos altos en regiones faciales no ocluidas y pesos bajos a las

que se activan con zonas ocluidas y de esta forma ignora las distorsiones de la oclusión.

MaskNet puede optimizarse con entrenamiento de extremo a extremo utilizando las etiquetas de identidades de las personas sin la necesidad de anotaciones adicionales. Toma una imagen de tamaño fijo como entrada, seguida de una CNN de poca profundidad ya que tienen un campo receptivo grande, mientras que para ser CNN profunda carece de descripción y hace difícil el aprendizaje de máscaras razonables. Hace uso de la función sigmoidea en lugar de la ReLU que es comúnmente usada y asigna valores en el intervalo de *[0,1]* para el normalizar el coeficiente de ponderación.

Dentro de la experimentación se usó la red Maxout (Goodfellow et al., 2013) de 8 capas convolución y 3 de agrupación, aplicando Masknet en la tercera capa de agrupación de dimensiones *9x9*. Entrenan con el *dataset* CASIA-Webface (Yi et al., 2014) que contiene 10,575 personas y 494,414 imágenes faciales las cuales se alinean usando MTCNN (K. Zhang et al., 2016a) y redimensionándolas a *100x100* y se voltean horizontalmente para poder aumentar sus datos. MaskNet aprende la ubicación de la información útil de la imagen de entrada e ignora de manera eficiente los fondos u objetos que no se relacionan.

Se ajusta el modelo con el *dataset* AR (A Martinez & Benavente, 1998) para experimentar con diferentes tipos de oclusiones usando las primeras 21 configuraciones de un total de 26 para cada uno de los 126 individuos. Posteriormente, se visualizan las máscaras mapeadas como se puede observar en la figura 2.15 de las cuales las dos primeras filas corresponden a AR y la última fila corresponde a imágenes de internet.

Figura 2.15 Máscaras mapeadas (Wan & Chen, 2018).

Para la evaluación de los *datasets* con oclusiones sintéticas se generan imágenes nuevas a partir de LFW (G. B. Huang & Learned-Miller, 2014) colocando un cuadro

negro de *nxn* aleatoriamente en cada imagen, como se muestra en la figura 2.16. Se utiliza Maxout (Goodfellow et al., 2013) y Resnet (Kaiming He et al., 2016) y se les incluye MaskNet y se aplica la tercera capa de agrupación para Mask-Maxout y la segunda capa de agrupación para Mask-ResNet.

Se obtuvo una mejora en la precisión de la verificación con LFW (G. B. Huang & Learned-Miller, 2014) en un 3.4% de Mask-Maxout sobre Maxout y 1.4% de Mask-ResNet sobre ResNet con un bloque de 60x60 con un tiempo 1.2 ms para la extracción de características. Con AR (A Martinez & Benavente, 1998) se obtuvo una mejora de 2.6% de Mask-Maxout sobre Maxout y 3.0% de Mask-ResNet sobre ResNet. Los experimentos muestran que MaskNet puede distinguir de forma precisa las áreas faciales útiles de las partes que se encuentran ocluidas.

Figura 2.16 Test con oclusión sintética (Wan & Chen, 2018).

Efficient Masked Face Recognition Method During the Covid-19 Pandemic (Hariri, 2020)

En el artículo, el objetivo es verificar si la persona lleva mascarilla o no. Implementan el algoritmo de Bolsa de Características como capa de agrupación en las redes neuronales convolucionales para reducir el número de parámetros y clasificar las imágenes de rostros enmascarados y obtener las mejores características de la región de los ojos y de la frente.

Extraen las características usando la red VGG-16 (Simonyan & Zisserman, 2014) considerando los mapas de características de la última capa, ya que contiene más de 14 millones de imágenes y 1,000 clases. Utilizan la librería Dlib-ml (King, 2009) para detectar 68 puntos de referencia faciales, normalizan a *240x240* píxeles y particionan en 100 bloques de *24x24*. Los autores se enfocaron en el conjunto de datos de RMFRD (Wang et al., 2020) que considera imágenes de rostros reales con máscara para abordar la problemática de la pandemia. Mencionan que la mejor tasa de reconocimiento se obtuvo en la última capa convolucional con 60 palabras con un 91.3%.

Facial Temperature y Mask Detection (sixphere et al., 2020)

Los autores explican que usaron tecnologías *Open Source* y dispositivos sencillos como son la Raspberry Pi (RPi), una cámara común, una cámara IR, una pantalla táctil y una carcasa para realizar su diseño. Específicamente, programaron la *Raspberry Pi 4 Modelo B*, la *Raspberry Pi Camera Module v2* y la Adafruit MLX90640 para capturar el calor de las cosas, además de usar *Raspberry Pi Touch Display* para poder implementar una red de aprendizaje profundo en *OpenCV* con un banco de 20,000 imágenes para detectar la máscara en el rostro, obteniendo un buen resultado. Todo el sistema lo montaron en una carcasa que fue impresa en *3D* y finalmente el sistema fue puesto a prueba. Los resultados obtenidos en las pruebas utilizando una PC y la Raspberry se muestran en la tabla 2.14.

Tabla 2.14 Pruebas del sistema (sixphere et al., 2020).

Escenario de prueba	En pc	En RPi
Haciendo uso de fase-api.js con SSD MobileNet	70 ms	8,000 ms
Haciendo uso de fase.api.js con Tiny	30 ms	1,500 ms
Haciendo uso de OpenCV.js con SSD ResNet10	320 ms	2,200 ms
Haciendo uso de OpenCV y tflite en Python	-	270 ms

Face Occlusion Detection Based on SSD Algorithm (Ziwei et al., 2020)

Se trabajó con el algoritmo de aprendizaje profundo SSD para clasificar y localizar oclusiones faciales, además de construir un *dataset* que consta de 5,252 imágenes las cuales fueron recopiladas manualmente considerando 7 tipos comunes de oclusiones como son: lentes de sol, bufandas, respiradores, gorros, anteojos, máscaras y collares; de las cuales 4,464 son utilizadas en el entrenamiento y 788 forman el conjunto de pruebas. Las imágenes fueron etiquetadas usando el software *3D labelling*.

Se utiliza VGG-16 para diseñar una estructura auxiliar para detectar características de múltiple escala disminuyendo el mapa de características gradualmente de *38x38* a *1x1* y un predictor de convolución usando un núcleo convolucional de *canal p* de *3x3* como elemento de predicción básico. El modelo de red VGG-16 se entrenó con el *dataset* ImageNet para la extracción de características y posteriormente se entrena con el *dataset* propio. Los parámetros establecidos fueron: lote de 16, un impulso de *0.9*, época cálida en 5, caída de peso en *0.0005*, tasa de aprendizaje inicial en *0.0001* y tasa de disminución de tasa inicial en *0.1* con un decaimiento de

tres ciclos de *80*, *40* y *180* respectivamente. Una vez entrenado el modelo, los resultados logrados se pueden apreciar en la tabla 2.15.

Tabla 2.15 Resultados de detección (Ziwei et al., 2020).

	Sunglases	Scarves	Respirator	Hats	Glass	Mask	Necklaces	Mean
AP/%	90.85	98.38	99.72	99.34	90.91	100	89	95.46

El modelo SSD se basa en la red neuronal convolucional Feedfordward para la detección del objetivo. El modelo SSD fue evaluado en cuanto a la calidad de detección de la oclusión facial con AP (precisión media) y mAP (precisión media media) y se puede apreciar que el método propuesto detecta y clasifica los 7 tipos comunes de oclusiones faciales con una precisión media de 95.46%; logrando hasta un 100% en la detección de la máscara. Los resultados anteriores se obtuvieron a pesar de que la imagen de prueba tiene un fondo complejo, el modelo SSD aun logra clasificar y ubicar con gran precisión la oclusión ver figura 2.17.

Figura 2.17 Resultado de detección (Ziwei et al., 2020).

Aplicación de Deep Learning para el Reconocimiento Facial con la Presencia de Oclusiones en el Contexto de la Pandemia Covid 2021 (Mucha et al., 2021)

Desarrollaron un sistema el cual aplica Deep Learning para reconocimiento facial aun cuando el rostro presenta mascarilla facial o lentes. Utilizan Python como software de desarrollo del sistema y como red convolucional pre entrenada VGG16. El sistema realiza el reconocimiento en 4 casos: con el rostro descubierto, con presencia de mascarilla quirúrgica, con el uso de lentes y con el uso de mascarilla quirúrgica y lentes.

Además, crearon un *dataset* que contiene 2,400 imágenes enfocando el rostro en diversos ángulos, pero sin perder los rasgos de la nariz, boca y la barbilla de las personas; además algunos rostros presentan mascarilla y/o lentes.

Haciendo uso de la red Haar cascade frontalface y la librería de numpy en Python identificaron las características y patrones de la imagen al momento de la activación de la webcam. Para el entrenamiento de la CNN alojaron el *dataset* en Google drive para facilitar el proceso, el entrenamiento se realizó en 500 épocas en un tiempo aproximado de 8 horas, el cual al finalizar proporcionó la CNN entrenada a la que nombraron Red.h5.

Realizaron pruebas finales con 5 personas con un tiempo de 10 segundos, arrojando los resultados mostrados en la tabla 2.16, con un porcentaje total de 71%. En la evaluación se pueden observar los siguientes casos sin lentes-sin mascarilla (SS), sin lentes-con mascarilla (SC), con lentes-sin mascarilla (CS) y con lentes-con mascarilla (CC). En este caso, la presencia de mascarilla y lentes tiene el menor desempeño.

Tabla 2.16 Resultados de la evaluación del sistema (Mucha et al., 2021).

CLASE	**TIPOS**				**PORCENTAJE**
	SS	SC	CS	CC	
Roy Uscamayta	100%	90%	80%	60%	82.5%
Jemima Elías	100%	80%	100%	70%	87.5%
Angiela Rojas	100%	80%	70%	40%	72.5%
Milagros Alfaro	40%	40%	30%	20%	32.5%
Luis Salazar	90%	80%	80%	70%	80%
PORCENTAJE TOTAL					71%

Mencionan que el tiempo de procesamiento depende de la cantidad de imágenes y de los ciclos para el entrenamiento, así mismo mencionan que mientras más imágenes con rotación de la cabeza se tengan, el reconocimiento mejora.

Detección Automática de Rostros con Cubreboca o sin Cubreboca para Restringir el Acceso a Institución Educativa (Cayetano, 2021)

Desarrollaron un sistema, basado en Deep Learning, para la detección automática del uso del cubrebocas en la entrada de una institución educativa de tal forma que

se permita o niegue el ingreso a la misma. Y crearon un d*ataset* propio con imágenes de personas con y sin el cubrebocas, con un total de 700 fotografías.

Dicho *dataset* fue creado con imágenes obtenidas de internet y tomadas por los autores en las cercanías de donde se desarrolló el sistema, tomando en cuenta diversidad de edad, tonalidades de piel y cambios de iluminación conforme a la hora del día en que se capturaron las imágenes y que se portara o no el cubrebocas.

El *dataset* se fragmenta en dos conjuntos: uno para el entrenamiento con 560 imágenes y para validación con 140 imágenes. Se trabajó en Python debido a la gran cantidad de librerías que soporta como lo es TensorFlow y keras.

Se crea un aumento del número de datos de entrenamiento por medio de zoom, escalado y rotación. Lo que permite crear variaciones de las imágenes para tener una mejor habilidad en los modelos de ajuste en el aprendizaje de nuevas imágenes, lo que a su vez reduce la sobrecarga de memoria, pero tarda más en entrenarse el modelo.

Cada capa convolucional está formada por *32, 64* y *128* filtros o conjunto de kernels esto es equivalente con la cantidad de matrices de salida en la capa oculta, las cuales se vuelven datos de entrada para las capas de *Maxpooling* haciendo a su vez uso de la función *Flatten* para redimensionar las matrices de entrada. Posteriormente, se clasifica cada imagen con ayuda de un filtro de 512 neuronas a las que se les suma la función de activación *ReLu*, desactivando el 50% de las neuronas para evitar el desbordamiento. Después, se utiliza un segundo filtro de clasificación de 256 neuronas desactivando a su vez el 50 % de ellas y de igual manera se les suma la función *ReLu*, de esta forma no se sobrecarga la memoria y permite que la red neuronal tome diversos caminos para obtener el mismo resultado de *ReLu.* Finalmente, en la capa de salida se obtienen las salidas binarias donde el 1 indica la pertenencia a cierta clase y el 0 en caso contrario; además, con ayuda de *SoftMax* se crea un vector de probabilidad que corresponde a la pertenencia de la clase en particular.

El sistema se evaluó con imágenes de personas diferentes a las del *dataset* de entrenamiento, logrando un 100% de exactitud. Indicando con un recuadro de color verde si portaba el cubrebocas y uno de color rojo si no lo portaba. También se evaluó la escala y debido a la lejanía de la persona, con respecto de la cámara, se obtuvo un 96% de exactitud ver figura 2.18.

Figura 2.18 Resultado de la evaluación de una persona ajena al *dataset* de entrenamiento (Cayetano, 2021).

2.2.4 Herramientas

Una Revisión Sistemática de Métodos para Localizar Automáticamente Objetos en Imágenes (Chaves et al., 2018)

En la actualidad la localización de objetos mediante visión artificial se ha vuelto de suma importancia en diferentes ámbitos como son los vehículos autónomos, la imagen médica, la inspección visual en la industria y la robótica que se clasifican obteniendo un conjunto de regiones en una imagen indicando si contiene o no el objeto de interés.

El artículo presenta una revisión de métodos que permiten localizar objetos de interés con algoritmos clásicos hasta los más recientes. Los autores agruparon en tres categorías:

- Ventana deslizante: La localización se realiza utilizando una ventana de tamaño variable, pero cambios en la iluminación y oclusión del rostro afectan una correcta detección de los objetos, entre ellos se encuentran:
  - HOG+SVM: HOG (Dalal & Triggs, 2005), SVM (Boser et al., 1992)
  - DPM (Felzenszwalb et al., 2010)
  - ESS (Lampert et al., 2008)
- Regiones candidatas: Se utilizan ventanas utilizando clasificadores más complejos que generan conjuntos de regiones candidatas, se encuentran:
  - OBJECTNESS (Alexe et al., 2010)
  - SELECTIVE SEARCH (Uijlings et al., 2013)
  - EDGE BOXES (Zitnick & Dollár, 2014)
- Aprendizaje profundo: Los métodos de aprendizaje profundo describen a los objetos directamente considerando niveles de abstracción diferentes. Algunos ejemplos son.
  - RCNN (R. Girshick et al., 2013)
  - FAST-RCNN (R. B. Girshick, 2015)
  - FASTER-RCNN (Ren et al., 2015)

- YOLO (Redmon et al., 2015)
- SSD (Liu et al., 2016)
- MASK-RCNN (K He et al., 2017)

En el artículo se realizó la revisión de más de 50 trabajos sobre la localización de objetos en imágenes digitales de forma automática, comentando que las estrategias basadas en ventanas deslizantes son las más costosas en términos computacionales seguidos de los métodos de regiones candidatas y por último los métodos de aprendizaje profundo.

Raspberry Pi and Computers-Based Face Detection and Recognition System (Wazwaz et al., 2018)

En el artículo se propone el desarrollo de un sistema de detección y reconocimiento de personas mediante la implementación una red de microcomputadoras y una Raspberry pi por medio de la conexión ethernet TCP / IP. El sistema captura el video por medio de la cámara conectada a la Raspberry para, posteriormente, ejecutar el algoritmo BCOSF[7] para la detección de los rostros en cada fotograma del video capturado y con el algoritmo LBPH[8] ecualizar la imagen facial para reducir las variaciones y comparar el histograma resultante con los que se encuentran en la base de datos. Si esa persona cuenta con varias imágenes se podrá reconocer con un mejor porcentaje de correlación, pero se tendrá un mayor coste en procesamiento y reconocimiento. Para optimizar el proceso de reconocimiento y detección se implementó el uso de tres servidores o microcomputadoras como se puede ver la figura 2.19.

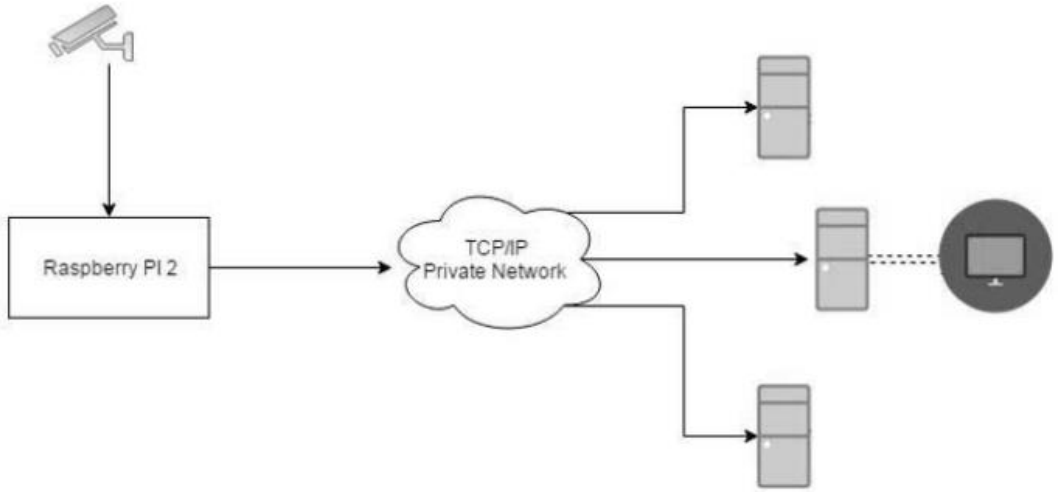


Figura 2.19 Diagrama de bloques del sistema general (Wazwaz et al., 2018).

7 https://docs.opencv.org/3.3.0/d7/d8b/tutorial_py_face_detection.html

8 https://docs.opencv.org/2.4/modules/contrib/doc/facerec/facerec_tutor%20ial.html#local-binary-patterns-histograms-in-opencv

Dentro de las pruebas que se realizaron para medir el rendimiento se observó que si se reduce el tamaño de la imagen disminuye la calidad y por consecuencia aumenta la tasa de error; destacando que el desempeño también varía según las condiciones como la iluminación, la distancia y la resolución de la cámara. Con respecto al aumento en el número de imágenes no se tenía retraso en la transmisión ya que son de tamaño pequeño y la velocidad de transmisión es alta de hasta 100Mbps.

Dentro de las características del conjunto de datos se destaca que contiene 100 categorías de los cuales algunos de los usuarios tienen 3 imágenes y otros tienen 25, de un total de 1,000 imágenes que integran el *dataset.*

Herta Lanza un Reconocimiento Facial que Permite Identificar Hasta con Mascarilla (Herta Lanza Un Reconocimiento Facial Que Permite Identificar Hasta Con Mascarilla, 2020)

Derivado de la pandemia mundial causada por el Covid-19 la empresa líder a nivel mundial de reconocimiento facial en multitudes, aceleró el desarrollo de un software que ayude a la identificación de personas con tecnología Deep Learning. La aplicación proporciona tasas de identificación altas aun cuando se oculte gran parte del rostro de las personas, haciendo uso de la zona más diferencial del rostro que se encuentra en los ojos, ver figura 2.20.

La implementación de este software permitirá la identificación en controles de fronteras sin quitarse la mascarilla y así evitar contagios mientras se espera en los centros de control. Además, buscan implementar el sistema de forma masiva en eventos, estadios deportivos, salud, gobierno y transporte. Mencionan que por su versatilidad escalable y compatibilidad con cualquier cámara IP el software es accesible para cualquier organización empresarial.

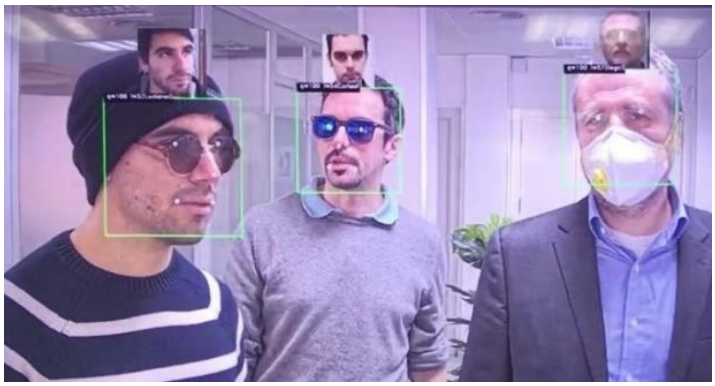


Figura 2.20 Ejemplo de funcionamiento del software (Herta Lanza Un Reconocimiento Facial Que Permite Identificar Hasta Con Mascarilla, 2020).

Algoritmo para la Detección de Mascarilla (J. Barrios, 2020)

El autor desarrolló un algoritmo para la detección del uso de la mascarilla, en tiempo real, con la versatilidad de poder instalar el algoritmo en una PC y usar una cámara web convencional para comprobar el uso del cubrebocas en la entrada de los sitios públicos, ver figura 2.21.

El algoritmo emplea herramientas de aprendizaje automático supervisado y *Deep Learning* donde se usan redes neuronales convolucionales, enseñando primeramente al algoritmo a detectar el uso de mascarilla con un conjunto de 3,835 imágenes de entre las cuales 1,916 son imágenes con mascarilla y 1,919 sin ella. Utilizaron herramientas tales como OpenCV, Keras y TensorFlow en Python.

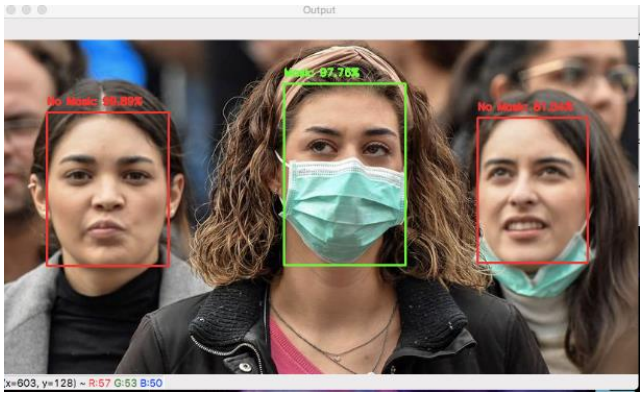


Figura 2.21 Implementación del algoritmo de detección (J. Barrios, 2020).

Face Recognition for Automatic Vehicle Ignition Based on Raspberry Pi (Nuñez & Nuñez, 2020)

El reconocimiento facial es parte importante en la seguridad de los vehículos para prevenir el robo de los mismos, por ello los autores desarrollaron una aplicación de reconocimiento facial para el encendido de los vehículos haciendo uso de una Raspberry pi y de OpenCV, principalmente para mejorar la seguridad de los automóviles y evitar el robo con diversas formas de encendido corruptas. La aplicación consiste en tres procesos principales: la detección del rostro, la recopilación de los datos y por último el entrenamiento del sistema.

Para poder realizar el reconocimiento facial se implementó el algoritmo AdaBoost gracias a las funciones de tipo Haar que incluye y así se calculan las características de un rostro típico y haciendo uso de una Raspberry Pi 4 Modelo B que cuenta con 40 pines de entrada/salida y una memoria máxima de 4gb de RAM y conectividad wifi de 2.4 y 5 GHz.

Las imágenes se adquieren mediante una cámara web y se almacenan para poder comparar los rostros detectados con los que se encuentran en los registros, posteriormente, se extraen las características únicas de cada rostro y crea un archivo binario con extensión *yml*. Por último, se compara la información del archivo *yml* que se generó con los registrados en el sistema y si coinciden se concede el acceso y se manda una señal por medio de los pines con un alto lógico a la computadora del vehículo, de lo contario se niega el acceso mandando un bajo lógico a la computadora del vehículo como se puede observar en la figura 2.22.

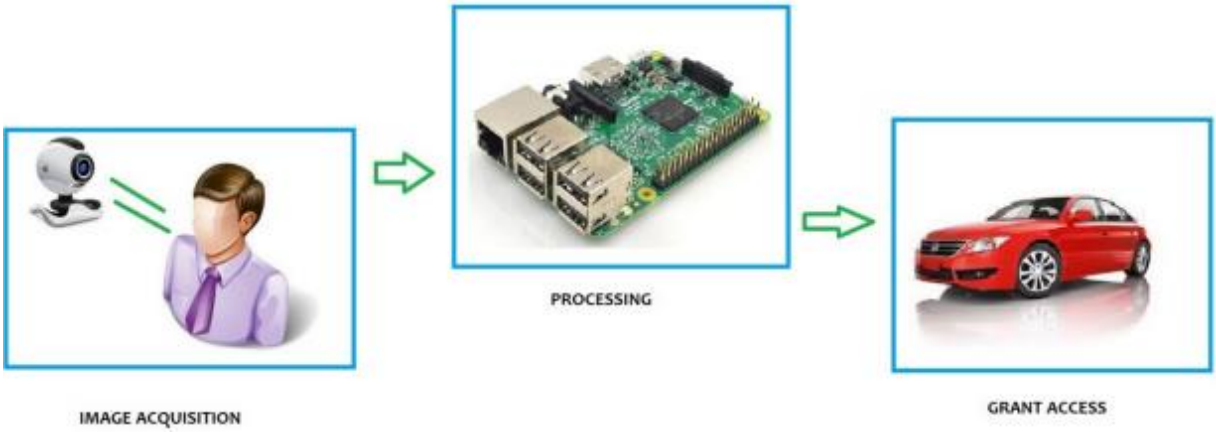


Figura 2.22 Arquitectura del proyecto (Nuñez & Nuñez, 2020).

El sistema comprobó la aplicación con el rostro de un usuario no registrado en el sistema y con un usuario simulando ser el propietario del vehículo al cual se le concede el acceso como se puede ver en la figura 2.23. El sistema muestra el nombre de la persona identificada y el porcentaje de coincidencia, en otro caso, se le asigna la etiqueta de desconocido.

Figura 2.23 Funcionamiento de la aplicación (Nuñez & Nuñez, 2020).

Con esta aplicación se ayuda a reducir costes de los seguros al aumentar la seguridad antirrobos de los vehículos y permitiendo guardar las imágenes de los delincuentes cuando intenten violar la seguridad de un vehículo.

2.2.5 Conjuntos de Imágenes

Detecting Masked Faces in the Wild with LLE-CNNs (Ge et al., 2017)

La detección de rostros con oclusiones se ha vuelto una tarea desafiante principalmente por dos razones: la ausencia de grandes *datasets* de rostros enmascarados y la ausencia de señales faciales de las regiones enmascaradas. Como forma de abordar estos problemas presentan un *dataset* denominado MAFA el cual consta de 30,811 imágenes de internet y 35,805 rostros enmascarados con diversas orientaciones y grados de oclusión, ver figura 2.24. Además, proponen el uso de LLE-CNN para la detección de rostros enmascarados recuperando señales faciales faltantes y suprimiendo la información no facial. Consta de tres módulos:

- *Propuesta*: Extrae y caracteriza a los candidatos faciales por medio de un descriptor 4096d con un modelo VGG-Face previamente entrenado (Parkhi et al., 2015)
- *Incrustación*: Recupera las señales faciales faltantes y suprime las características ruidosas incorporadas por las regiones de la máscara.
- *Verificación*: clasifica a los candidatos faciales a partir de las señales faciales refinadas y refina sus posiciones y escalas

Figura 2.24 Los rostros enmascarados pueden tener diferentes orientaciones, grados de oclusión y tipos de máscaras (Ge et al., 2017).

Recopilaron imágenes faciales de internet utilizando palabras claves como cara, máscara y oclusión logrando recuperar más de 300,000 imágenes de redes sociales como Flickr y motores de búsqueda como Google y Bing para la construcción del *dataset* MAFA. Además, definieron seis atributos para identificar las caras en las imágenes obtenidas, utilizando las ubicaciones de: las caras, los ojos y las mascarillas. Aparte de la información facial consideraron la orientación, el grado de

oclusión y el tipo de máscara. En la figura 2.25 se puede apreciar un ejemplo de identificación de cara con los atributos antes mencionados.

En los resultados experimentales, los autores mencionan superar con al menos un 15.6% a los últimos 6 avances relacionados con el *dataset* MAFA. Concluyen que el enfoque propuesto funciona en el ambiente real, que siempre será desafiante por las condiciones cambiantes de iluminación o de oclusiones inesperadas, pero son de suma importancia en diversas aplicaciones como la videovigilancia.

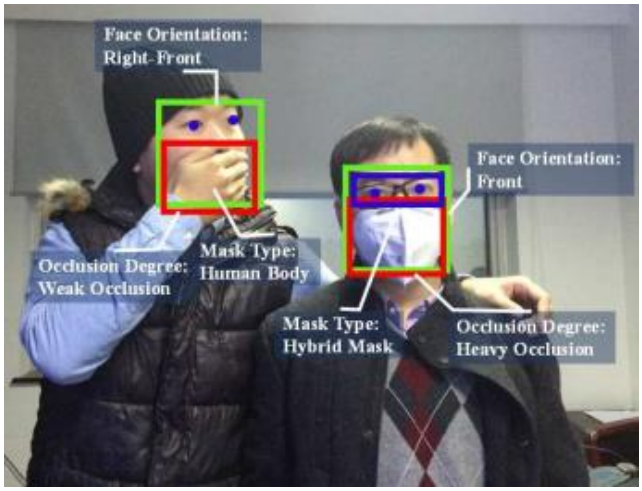


Figura 2.25 Ejemplos de caras identificadas en MAFA (Ge et al., 2017).

VGGFace2: A Dataset for Recognizing Faces Across Pose and Age (Cao et al., 2018)

Se realizó la creación de un *dataset* a gran escala denominado VGGFace2, el cual contiene 3.31 millones de imágenes de 9,131 sujetos todas ellas obtenidas del motor de búsqueda de imágenes de Google con variaciones de pose, edad, iluminación, etnia y profesión. Se buscó tener una gran cantidad de identidades de figuras públicas (para cada clase) para minimizar el ruido.

El conjunto de datos tiene aproximadamente un equilibrio de género con un 59.3% de hombres que varía entre 80 y 843 imágenes para cada persona. El *dataset* está dividido en dos grupos uno para entrenamiento con 8,631 clases y otro con 500 clases para la evaluación.

Para la detección del rostro se implementaron hiperparámetros con el modelo proporcionado por (K. Zhang et al., 2016a) para favorecer una buena compensación entre la precisión y la recuperación. Se amplió el cuadro delimitador de la cara con un factor de *0.3* para incluir toda la zona de la cabeza y obtener las características faciales. Finalmente, se obtiene un descriptor de las redes entrenadas cambiando el tamaño del cuadro delimitador de la cara a 256 píxeles del lado más corto,

posteriormente se recorta de forma central y se clasifica mediante la función de pérdida *soft-max*. Se implementa el modelo VGGFace (Parkhi et al., 2015) (Parkhi et al., 2015) y para la eliminación de las imágenes casi duplicadas se agruparon los descriptores VLAD (Arandjelovic & Zisserman, 2013).

Se realiza un filtrado manual para conseguir una pureza superior al 96% ya que algunos sujetos contienen imágenes con parejas u otros miembros y generan ruido. Para obtener información sobre la pose y la edad se entrenó el clasificador Resnet-50 (Kaiming He et al., 2016) de 5 vías con el conjunto de imágenes CASIA-WebFace (Yi et al., 2014), y se obtiene un vector plantilla promediando los descriptores faciales y los clasificadores SVM implementados para la identificación.

Para 500 sujetos en el conjunto de evaluación se eligieron a 50 imágenes al azar para prueba y el resto como entrenamiento aprendiendo los clasificadores de SVM de 1 contra el resto. Se evaluaron tres vistas: frontal, tres cuartos y de perfil y, se pudo concluir que los modelos funcionan mejor cuando se combinan poses similares a diferencia de cuando se presentan diferentes, lo que demuestra que el problema de la pose es más difícil.

En la tabla 2.17 se muestra la comparación entre diferentes conjuntos de entrenamiento y modelos entrenados en la propuesta.

La superioridad de SENet (Hu et al., 2018) sobre ResNet-50 (Kaiming He et al., 2016) es evidente tanto en la identificación como en la verificación, con las configuraciones de entrenamiento.

Tabla 2.17 Comparativa entre diferentes conjuntos de entrenamiento (Cao et al., 2018).

Training dataset	Arch.	1:1 Verification TAR				1: N Identification TPIR				
		FAR= 1E−5	FAR= 1E−4	FAR= 1E−3	FAR= 1E−2	FPIR= 0.01	FPIR= 0.1	Rank-1	Rank-5	Rank-10
VGGFace [16]	ResNet-50	0.342	0.535	0.711	0.85	0.429 ± 0.024	0.635 ± 0.015	0.752 ± 0.038	0.843± 0.032	0.874± 0.026
MS1M [7]	ResNet-50	0.548	0.743	0.857	0.935	0.662± 0.036	0.810± 0.028	0.865± 0.053	0.917± 0.032	0.936± 0.024
VGGFace2	ResNet-50	0.647	0.784	0.878	0.938	0.701± 0.038	0.824± 0.034	0.886± 0.032	0.936± 0.019	0.953± 0.013
VGGFace2_ft	ResNet-50	0.671	0.804	0.891	0.947	0.702± 0.041	0.843± 0.032	0.894± 0.039	0.940± 0.022	0.954± 0.016
VGGFace2	SENet	0.671	0.8	0.888	0.949	0.706± 0.047	0.839± 0.035	0.901± 0.030	0.945± 0.016	0.958± 0.010
VGGFace2_ft	SENet	**0.705**	**0.831**	**0.908**	**0.956**	**0.743± 0.037**	**0.863± 0.032**	**0.902± 0.036**	**0.946± 0.022**	**0.959±0.015**
Whitelam et al. [22]	-	0.350	0.540	0.700	0.840	0.420	0.640	0.790	0.850	0.900

Masked Face Recognition Datasets and Validation (B. Huang et al., 2021)

Se presenta un sistema para el reconocimiento de rostros con máscara. También propusieron tres *datasets* de rostros enmascarados: MFDD que contiene 24,771 imágenes; RMFRD que contiene 4,015 imágenes de 426 personas que se organizan en 7,178 pares de muestras enmascaradas y desenmascaradas y SMFRD que contiene 536,721 imágenes de 16,817 personas los cuales se menciona fueron construidos por diferentes medios. Mencionan que los conjuntos son para utilizarse en dos aplicaciones: para la detección de la mascarilla facial y el reconocimiento de rostros enmascarados.

Utilizaron RetinaFace para detectar y alinear las imágenes de rostros que no fueron procesados y los normalizan a *112x112*. Realizan un aumento en el *dataset* de entrenamiento para mejorarlo, rotando las imágenes haciendo uso de Py-torch, con una tasa de aprendizaje de *0.1* con un total de 34 épocas finales. Mencionan que utilizaron ArcFace para el reconocimiento facial adoptando por separado ResNet18, ResNet34, ResNet50 y ResNet100 como referencia de evaluación de otros investigadores que hacen uso de los *datasets* propuestos.

Dentro de las métricas de evaluación usaron la verificación *1:1* para el reconocimiento facial, para ICCV2021-MFR se mide la tasa de aceptación real (TAR) con el protocolo máscara a no máscara *1:1*, con tasa de falsa aceptación (FAR) menor a 10^{-4}, en el caso de otros *datasets* TAR se mide *1:1* y FAR de 10^{-6}.

Para el entrenamiento usaron los conjuntos WebFace, SMFRD, MS1MV3 y Glint360K, y para la etapa de pruebas, evaluaron los modelos con RMFRD e ICCV2021-MFR integrados, ICCV2021-MFR-MASK que contiene 6,964 imágenes enmascaradas y 13,928 no enmascaradas, ICCV2021-MFR-ALL como conjunto Multirracial con 1,624,305 imágenes de 242,143 personas.

En la verificación del rostro comparan el desempeño del sistema ante los *datasets*: RM-FRD, ICCV2021-MFR-MASK e ICCV2021-MFR-ALL. Como se puede ver en la tabla 2.18, se observa que ArcFace entrenado con el conjunto SMFRD supera el modelo obtenido con el conjunto WebFace en RMFD. En general, ResNet100 entrenado tiene mejor desempeño con los diferentes *datasets* de prueba y también mencionan que la precisión de varios *dataset* los modelos en RMFD se correlacionan positivamente con el rendimiento en ICCV2021-MFR-MASK.

Tabla 2.18 Comparación de la verificación del rostro (B. Huang et al., 2021).

Dataset	Backbone	Method	Size/MB	RMFRD	ICCV2021-MFR-MASK	ICCV2021-MFR-ALL
WebFace	R50	Arcface	166	63.22	22.13	38.51
SMFRD	R50	Arcface	166	71.13	39.03	39.16
MSIMV3	R18	Arcface	91	64.19	47.85	68.33
Glint360K	R18	Arcface	91	64.59	53.32	72.07
MSIMV3	R34	Arcface	130	65.90	58.72	77.36
Glint360K	R34	Arcface	130	68.61	65.10	83.02
MSIMV3	R50	Arcface	166	68.68	63.85	80.53
Glint360K	R50	Arcface	166	71.40	70.23	87.08
MSIMV3	R100	Arcface	248	70.19	69.09	84.31
Glint360K	R100	Arcface	248	72.74	75.57	90.66

MaskedFace-Net – A Dataset of Correctly/Incorrectly Masked Face Images in the Context of COVID-19 (Cabani et al., 2021)

A inicios del 2020, la organización mundial de la salud (OMS) recomendó el uso obligatorio de la mascarilla o cubrebocas (como es conocido en México) como solución para limitar la propagación de COVID-19 ante la pandemia mundial, aumentando su demanda considerablemente. Sin embargo, muchas personas en el mundo no hacen uso correcto de la mascarilla por mal comportamiento o vulnerabilidad como puede ser en el caso de niños y ancianos. Los autores, como medida de contribución a las campañas de concientización del uso correcto del mismo, crearon el conjunto de imágenes llamado MaskedFace-Net

El objetivo del desarrollo del conjunto de imágenes MaskedFace-Net es presentar el modelo de máscara-rostro para: a) la identificación de rostros de personas con mascarilla y sin mascarilla y b) identificar rostros con sus mascarillas correctamente puestas o incorrectamente colocadas (sin cubrir nariz, en la barbilla o sin cubrir la barbilla) en áreas concurridas y reguladas.

MaskedFace-Net está creada tomando como base en el conjunto de imágenes faciales Flickr-Faces-HQ3[9] por su gran variedad de términos de edad, etnia, punto de vista, iluminación y fondo de imagen. Para su creación utilizan la metodología mostrada en la figura 2.26. Es decir, a partir de la imagen, aplican clasificadores en cascada con características Haar para detección de áreas de interés y ponen de manera artificial la mascarilla. El *dataset* cuenta con más de 137,016 imágenes de rostros enmascarados.

9 “dataset of face images Flickr-Faces-HQ (FFHQ)” https://github.com/NVlabs/ffhq-dataset.

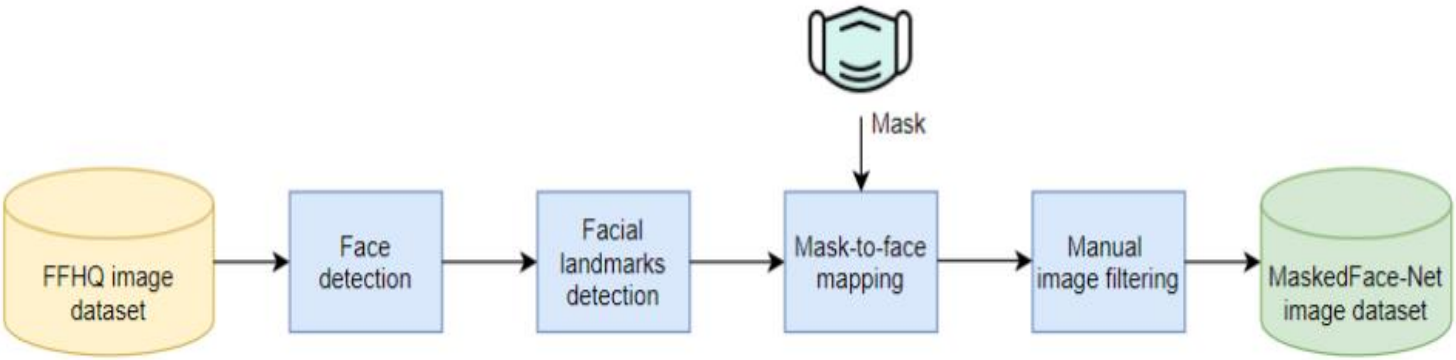


Figura 2.26 Diagrama del enfoque de edición de imágenes aplicado para generar el conjunto de datos de imágenes faciales enmascaradas correcta / incorrectamente "MaskedFace-Net" (Cabani et al., 2021).

El *dataset* se integra de tres subconjuntos, ejemplos de las imágenes se muestran en la figura 2.27:

- El conjunto de rostros correctamente enmascarados (cubriendo la nariz y boca) (CMFD por sus siglas en inglés), que integra el 49% de imágenes.
- El conjunto imágenes de rostros incorrectamente enmascarados (IMFD por sus siglas en inglés) corresponde al 51%.
- La combinación de (CMFD) y (IMFD) para la detección global de rostros enmascarados (MaskedFace-Net).

(a) (b)

Figura 2.27 Ejemplos de imágenes del dataset MaskedFace-Net (a) Ejemplo del conjunto de rostros correctamente enmascarados (CMFD). (b) Ejemplo del conjunto de rostros incorrectamente enmascarados (IMFD), (Cabani et al., 2021).

2.2.6 Uso de la Mascarilla

Consejos para la Población Sobre el Nuevo Coronavirus (2019-nCoV): Cuándo y Cómo Usar Mascarilla (ONU, 2020).

Convertir el uso de la mascarilla es una parte normal de su interacción con otras personas adoptando algunas precauciones sencillas. Para que sean lo más eficaces posibles, es esencial utilizar, guardar, limpiar y eliminar las mascarillas correctamente. Indicaciones básicas sobre la manera de ponerse la mascarilla:

- Lávese las manos antes de ponerse la mascarilla, y también antes y después de quitársela, y cada vez que la toque.
- Compruebe que le cubre la nariz, la boca y el mentón.
- Cuando se quite la mascarilla, guárdela en una bolsa de plástico limpia; si es de tela lávela cada día y si es una mascarilla médica, tírela a un cubo de basura.
- No utilice mascarillas con válvulas.

Uso Generalizado de Cubrebocas Frente a la Pandemia Ocasionada por el Sars-CoV-2 (López Ortiz & López Ortiz, 2020)

Mencionan que el uso generalizado del cubrebocas puede salvar vidas frente a la pandemia del COVID-19 y reducir los contagios entre la población, menciona que cerca de la mitad de quienes lo portan lo hacen de manera incorrecta al cubrir únicamente la boca o lo portan a nivel del mentón o del cuello.

A continuación, la tabla 2.19, muestra una síntesis de los artículos revisados.

Tabla 2.19 Resumen de artículos del estado del arte.

Artículo	Objetivo	Técnicas/ Herramientas	Conjuntos	Resultados
Reconocimiento de Rostros con Oclusión				
(Azeem et al., 2014)	Realizó una recopilación de los principales métodos utilizados en la biometría del rostro en presencia de oclusión.	Métodos basados en características, basados en piezas y basados en fractales.	-	Los métodos basados en partes que después son analizados en bloques individuales son los más utilizados.
(Cornejo & Pedrini, 2016)	Se trabajó el reconocimiento de emociones resistentes a la oclusión mediante características de histograma.	RPCA, Seguimiento de rostro y ojos de Chera, LBP, CENTRIS, PCA, LDA, KNN, SVM.	Cohn-Kanade (CK+), JAFFE.	Compararon su enfoque propuesto (CENTRIS+ PCA+ LDA+ SVM) con otros métodos y logran mejores resultados para los *dataset* CK+ y JAFFE bajo oclusión.

Tabla 2.19 Resumen de artículos del estado del arte.

Artículo	Objetivo	Técnicas/ Herramientas	Conjuntos	Resultados
Reconocimiento de Rostros con Oclusión				
(Gonzalez-Sosa et al., 2016)	Proponen tratar la oclusión facial mediante la fusión de regiones del rostro no ocluido.	Face++, LBP, PCA.	ARFace.	Se observó que Face++ se comportó mejor que LBP en el escenario neutral a diferencia de la presencia de oclusiones donde LBP con 31% sobre 7.35% de EER de Face++.
(Zhang et al., 2018)	Propusieron un algoritmo para detectar la oclusión del rostro implementando la forma Omega que se forma entre la cabeza y los hombros de las personas para implementarlo en cajeros automáticos.	Clasificador de color de piel, clasificador de coincidencias, AdaBoost.	Conjunto de 120 videos.	De entre 120 videos se obtuvo: 97.95% para rostro normales, 93.57% para rostros ocluidos, 96.70% con el color de la cara, 94.35% usando la plantilla de la cara y aumenta hasta un 98.56% si se implementa un clasificador en cascada.
(Zhao, 2018)	Desarrollaron un método al que denominaron PLOD_FR para detectar parte de la oclusión a nivel píxel basado en representación dispersa.	SRC, WGSR.	AR, Yale B, PIE.	Se menciona que su modelo propuesto junto con WGSR logra altas tasas de reconocimiento en presencia de oclusiones, poses y presencia de lentes y bufanda.
Detección del Rostro				
(Garcia Rios et al., 2017)	Se propuso un sistema para el reconocimiento facial con un modelo wireframe y Facefit.	Facefit (Ajuste de rostro), Modelo Apariencia Activa, SVM, Librería LIBSVM.	900 imágenes estéreo de un total de 60 personas.	Se menciona que se obtiene un comportamiento favorable con valores arriba del 95% de reconocimiento.
(Singh & Prasad, 2018)	Se realizó una recopilación de técnicas para el reconocimiento facial en 2D y 3D.	HOG, Procesamiento de características, Clasificación de fusión RF, Estrategia multiescala basada en descriptores geométricos y locales, LBP, Modelo de forma, Característica geométrica local y coincidencia de forma, Descriptor facial basado en el filtro Gabor y adaptación de dominio.	ORL, YALE, PHPID, VLC, FRGC, LFW y UGC-JU.	Se menciona que de todos los datasets analizados se obtuvo una precisión de hasta 99.33 % con LFW y 99.07% con UGC-JU.
(Song et al., 2019)	Proponen una estrategia de aprendizaje de máscaras para poder identificar y descartar elementos que sean característicos para el reconocimiento facial.	red siamesa diferencial por pares (PDSN), modelos de CNN, descarte de características (FDM).	FacesCrub.	Protocolo 1: 99.72% con gafas de sol y 100% con bufanda. Protocolo 2:98.18% con gafas de sol y 98.33% con bufanda.

Tabla 2.19 Resumen de artículos del estado del arte.

Artículo	Objetivo	Técnicas/ Herramientas	Conjuntos	Resultados
Detección del Rostro				
(Cutipa et al., 2019)	Implementar la combinación de wavelets Gabor y PCA para la detección y reconocimiento automático.	SVM, PCA, Clasificador cascada.	FERET, ORL y Propio.	Se menciona que su sistema obtuvo un 98% de precisión.
(Wang et al., 2020)	Desarrollo de un modelo para el reconocimiento de personas con imágenes de rostros enmascarado.	Modelo de granularidad múltiple basado en ojos y el rostro, Dlib.	MFDD, RMFRD, SMFRD.	Alcanza una precisión del 95%.
(Golwalkar & Mehendale, 2020)	Detectar el uso de mascarilla tanto en imágenes como en videos de tiempo real.	OpenCV, Python, Deep learning, HOG.	LFW.	Se probó con 325 imágenes y 7 videos identificando al 100% una sola cara, pero fallo al intentar detectar una cara cuando había más de una persona con mascarilla y se encontraban presentes en el mismo marco.
(Wudan, 2020)	Se ha implementado el algoritmo LeewatHertz para asegurarse de que el personal cumpla con la norma de portar el cubrebocas.	algoritmos de postura y clasificación, LeewatHertz.	-	Mencionan que los algoritmos son 705 menos preciso para identificar caras nuevas cuando se entrenan con bases de datos estándar.
(Mundial et al., 2020)	Reconocimiento de rostros enmascarados con el clasificador SVM.	Aprendizaje automático, clasificador SVM.	VGGFACE2.	Con SVM obtuvieron hasta 97% y hasta 98% de precisión al incluir poses laterales del rostro.
(Montero et al., 2021)	Se abordó el problema del reconocimiento facial con mascarilla con un enfoque basado en ArcFace y un dataset de gemelos enmascarados.	LResNet50, MTArcFace, MaskTheFace.	MS1MV2, LFW, CFP, Agebd.	Se menciona que método propuesto supera ampliamente ArcFace y alcanza una precisión media de 99.78%.
(Loey et al., 2021)	Propuso un modelo híbrido que implementa el aprendizaje automático clásico y profundo para la detección de máscaras faciales.	SVM, Resnet50.	RMFD, SMFD, LFW.	El clasificador SVM logró en RMFD una precisión de prueba de 99.64%, en SMFD logró un 99.49% de precisión y en LFW logró 100%de precisión.

Tabla 2.19 Resumen de artículos del estado del arte.

Artículo	Objetivo	Técnicas/ Herramientas	Conjuntos	Resultados
Aprendizaje Profundo				
(Bu et al., 2017)	Realizan un marco en cascada basado en CNN que consta de tres redes para la detección de rostros enmascarados y propusieron el dataset MASKED FACE.	Clasificadores CNN.	MASKED FACE, WIDER FACE.	Se evaluó con PASCAL VOC obteniendo una precisión de 86.6% y 87.7% de precisión de recuperación.
(Wu et al., 2017)	Desarrollaron un sistema para reducir el tiempo de procesamiento con clases latentes creando grupos iniciales pequeños.	SIFT, K-medias, LDA, Muestreo de Gibbs, RCPR, LBP-I-RCPR.	COFW.	Se menciona que el método propuesto puede reducir el tiempo en un 65.84% con respecto a LBP-I-RCPR y mantiene la precisión de localización de puntos de referencia y la detección de oclusiones no se reduce.
(Li et al., 2018)	Proponen un modelo de red neuronal para reconocer expresiones faciales bajo oclusiones por medio de un mapa de características.	VGG 16.	RAF-DB, AffectNet.	Su modelo de red neuronal logró un 83.27 en el reconocimiento sin oclusión y 78.05 % con oclusión con RAF-DB y 55.33% sin oclusión y 52.47% con oclusión en AffectNet.
(Wan & Chen, 2018)	Se propusó y desarrollo un modelo entrenable llamado Masknet que se puede incluir en la arquitectura de CNN existentes.	Implementación de la función sigmoidea, MAxout, MTCNN, ResNet.	Casia-WebFace, AR, LFW.	Se obtuvo mejora de 3.4 con de Mask-Maxout sobre Maxout y 1.4% de Mask-ResNet sobre ResNet con LFW. Con AR se obtuvo una mejora de 2.6% de Mask-Maxout sobre Maxout y 3.0% de Mask-ResNet sobre ResNet.
(Hariri, 2020)	Comprueban si la persona lleva mascarilla o no para posteriormente, implementar el algoritmo de Bolsa de Características como capa de agrupación en CNNs.	VGG-16, Librería Dlib-ml.	RMFRD.	La mejor tasa de reconocimiento fue del 91.3%.
(sixphere et al., 2020)	Implementaron una red de aprendizaje profundo en *OpenCV* con un banco de 20,000 imágenes para detectar la máscara en el rostro.	OpenCV, Python.	Conjunto de 20,000 imágenes.	Implementando en PC se logró una detección en 70 ms mientras que en una Raspberry fue de 270 ms.

Tabla 2.19 Resumen de artículos del estado del arte.

Artículo	Objetivo	Técnicas/ Herramientas	Conjuntos	Resultados
Aprendizaje Profundo				
(Ziwei et al., 2020)	Se trabajó con el algoritmo de aprendizaje profundo SSD para clasificar y localizar oclusiones faciales y se construyó un dataset de 5,252 imágenes.	CNN FeedFordwared, VGG16, SSD.	5,252 imágenes recopiladas manualmente, ImageNet.	Se obtuvo un 95.46% de precisión media de diversos tipos de oclusión y hasta un 100% en la identificación de mascarilla.
(Mucha et al., 2021)	Se desarrolló un sistema para el reconocimiento facial con oclusión por mascarilla y lentes, con Deep Learning.	Python, VGG16, Haar cascade frontal face, NumPy. Red.h5.	2,400 imágenes.	Reportan un accuracy de 71% en la identificación de personas en un tiempo aproximado de 10 segundos por detección.
(Cayetano, 2021)	Se desarrolló un sistema basado en Deep Learning para la detección de cubrebocas y creación de un dataset de 700 imágenes.	Python, TensorFlow, Keras, ReLu, SoftMax.	700 imágenes propias.	96% de precisión en el reconocimiento del cubrebocas.
Herramientas				
(Chaves et al., 2018)	Presenta una revisión de métodos que permiten localizar objetos de interés con algoritmos clásicos hasta los más recientes.	HOG+SVM: HOG, DPM, ESS, OBJECTNESS, SELECTIVE SEARCH, EDGE BOXES, RCNN, FAST-RCNN, FASTER-RCNN, YOLO, SSD, MASK.	-	Estrategias basadas en ventanas deslizantes son las más costosas computacionalmente, después los métodos de regiones candidatas y por último los métodos de aprendizaje profundo.
(Wazwaz et al., 2018)	Implementar una red de microcomputadoras para optimizar el proceso de reconocimiento y detección.	Algoritmo BCOSF, algoritmo LBPH.	1,000 imágenes de 100 personas.	Si se reduce el tamaño de la imagen disminuye la calidad de la imagen y por consecuencia aumenta la tasa de error. El desempeño varía de acuerdo a las condiciones presentes como la iluminación, la distancia y la resolución de la cámara.
(Herta Lanza Un Reconocimiento Facial Que Permite Identificar Hasta Con Mascarilla, 2020)	Desarrollo de un software que ayude a la identificación implementado tecnología Deep Learning proporcionando tasas de identificación muy altas.	Deep learning.	-	Mencionan que los algoritmos de Herta proporcionan tasas muy altas de identificación.
(Barrios, 2020)	Desarrollo un algoritmo empleando técnicas de inteligencia artificial para la detección del uso de la mascarilla en tiempo real.	OpenCV, Keras, TensorFlow, Deep Learning.	Conjunto 3,835 imágenes.	Alcanzando un 96.7% al detectar la mascarilla y un 99.89% al no detectar la mascarilla.

Tabla 2.19 Resumen de artículos del estado del arte.

Artículo	Objetivo	Técnicas/ Herramientas	Conjuntos	Resultados
Herramientas				
(Nuñez & Nuñez, 2020)	Desarrollaron una aplicación de reconocimiento facial para el encendido de los vehículos.	AdaBoost, haar cascade, función Recognizer predict.	Imágenes propias de los autores.	Lograron obtener hasta un 51.0% de precisión en la detección de usuarios permitidos para encender el vehículo.
Conjuntos de Imágenes				
(Ge et al., 2017)	Presentan un *dataset* denominado MAFA el cual consta de 30,811 imágenes de internet y 35,805 rostros enmascarados.	VGG-Face.	Flickr, Google, Bing, MAFA.	Logró superar con al menos un 15.6% a los últimos 6 avances relacionados.
(Cao et al., 2018)	Crean un dataset de gran escala denominado VGGFace2 con ayuda de hiperparámetros para detectar el rostro.	Hiperparámetros, VGGFace, Descriptores VLAD, ResNet-50, SVM, soft-max, SENet.	Casia-WebFace, MS1M.	Se observó la superioridad de SENet sobre ResNet-50.
(B. Huang et al., 2021))	Se propusieron 3 datasets de rostros enmascarados y fueron puestos a prueba.	ArcFace, ResNet18, ResNet 34, ResNet50, ResNet100, RetinaFace, Py-torch.	MFDD, RMFRD, SMFRD, WebFace, MS1MV3, Glink360K, ICCV2021-MFR, ICCV2021-MFR-ALL.	Se menciona que ArcFace entrenado en SMFRD supera el entrenamiento con WebFace en RMFD, así también ResNet50 entrenado en Glint360K es un poco mejor que entrenado en SMFRD.
(Cabani et al., 2021)	Presentar el modelo de máscara-rostro para permitir la generación de otras imágenes de rostros enmascarados.	Clasificadores en cascada con características de Haar.	MaskedFace-Net.	MaskedFace-Net integra CMFD en 49% y IMFD en 51%.
Uso de la Mascarilla				
(ONU, 2020)	Proporciona información acerca del uso correcto del cubrebocas.	-	-	-
(López Ortiz & López Ortiz, 2020)	Proporciona información del uso generalizado del cubrebocas frente a la pandemia del COVID-19.	-	-	-

Comentarios:

Como puede verse, debido a la pandemia en el 2020 se propusieron varios desarrollos de sistemas que realizan la detección automática de si una persona

porta cubrebocas y si se encuentra bien puesto. Se han creado varios conjuntos de datos de imágenes reales y simuladas para poder entrenar los algoritmos de clasificación clásicos y de aprendizaje profundo.

2.3 Tesis Relacionadas

En el CENIDET se han realizado diversos trabajos de investigación considerando el rostro, por ejemplo, el reconocimiento de expresiones faciales, estados de ánimo y la identificación de las personas, algunos de ellos se mencionan a continuación.

a) "*Reconocimiento de rostros invariante a expresiones faciales*" (Magadán, 1999), la autora propuso una solución para la identificación de personas sin que afecte la expresión facial presente en el rostro. El sistema contempla algunos factores presentes en el mundo real como la escala y traslaciones. Implemento el algoritmo BT que se basa en conceptos de testor y testor típico para describir el rostro y clasifica mediante el algoritmo ALVOT. Obtuvo el 100% de reconocimiento con la expresión neutral; sin embargo, si están presentes las siguientes emociones, la identificación de la persona disminuyó, logrando un: 63% con alegría, 70% con enojo y 80% con tristeza; siendo la eficiencia total de 80%.
b) "*Seguimiento y caracterización de componentes del rostro para la detección de expresiones faciales*" (Soto, 2009), el autor propuso un sistema de localización de forma automática la cara, así como de sus componentes en una secuencia de imágenes; en este caso ojos, cejas y boca, para reconocer las expresiones faciales de: alegría, tristeza, enojo, miedo, sorpresa y neutral. Las pruebas del sistema incluyen desde la detección de piel en más de un espacio de color hasta la clasificación de imágenes con Redes Neuronales Artificiales y C5.0. El autor evaluó con un conjunto de 50 videos y 40 imágenes del conjunto MMI logrando una efectividad con C5.0 del 90% y 88% con la RNA.
c) "*Valoración automática de la motivación y atención del estudiante mediante expresiones faciales*" (Vázquez, 2017), en su trabajo, la autora, utilizó el *Kinect* y la librería *facetracking* para crear un sistema que localiza automáticamente el rostro y describe sus elementos para detectar la motivación de la persona en la realización de una actividad. Ella creó un conjunto de videos en donde participaron 20 personas de diferentes rasgos en la cara, tono de piel, sexo y tamaño de la cara a una distancia de entre 60 a 120cm de distancia del *Kinect* con iluminación natural. Finalmente, reporta una certeza de 89.79% para los casos donde se determinó que los sujetos prestaron atención y un 90.72% en los casos donde no prestaban atención.

d) “*Reconocimiento automático del rostro para verificación de identidad para evaluación en línea”* (Valderrama, 2019) La autora desarrolló un sistema de visión artificial que monitorea en tiempo real a un aspirante que presenta un examen de admisión en línea en un ambiente no controlado. Su metodología de solución se dividió en 6 fases: en la etapa de procesamiento se implementaron 5 algoritmos: ecualización de histograma, ecualización adaptativa, modelo retina, retinex multi escala y el quinto algoritmo es el modelo propuesto por la autora que es la fusión de los algoritmos de retina y retinex. Para localizar las partes relevantes el rostro (cara, ojos, boca, nariz, etc.) utilizó la librería Dlib y evaluó varios algoritmos de clasificación. Utilizó 3 bancos de imágenes: FEI Face-Database, Faces94 y el tercero es un banco de imágenes propio, el cual contiene imágenes de 20 personas con oclusión parcial. La autora reporta que el sistema es capaz de proporcionar una respuesta de verificación de identidad por debajo de los 0.3 segundos.

Como se observa, existe información y experiencia en la localización y descripción del rostro sin oclusión. Sin embargo, se consideró analizar el uso de algunos algoritmos y librerías mencionadas en dichas tesis para la descripción del rostro con oclusión.

Capítulo 3

Marco Teórico

3.1 Etapas de un Sistema de Visión Artificial

De acuerdo con (González & Woods, 1996) un Sistema de Visión Artificial (SVA) consta de cinco etapas, ver figura 3.1.

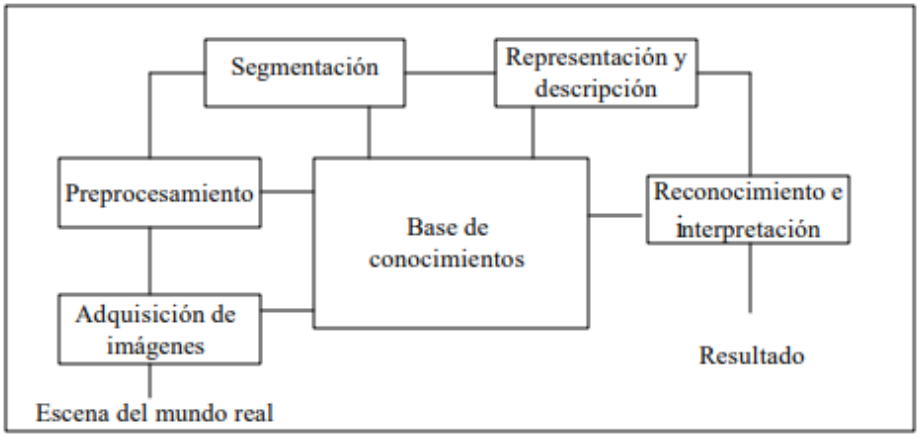


Figura 3.1 Módulos que constituyen un sistema de visión artificial (González & Woods, 1996).

Dichas etapas se desarrollan en el sistema implementado en la presente tesis y se estudiarán a detalle las técnicas utilizadas en cada fase.

3.2 Procesamiento de Imágenes

3.2.1 Modelos de Color

a) Modelo de Color HSV

De acuerdo con (González et al., 2009), el espacio de color HSV (tono, saturación, valor) es uno de varios sistemas de color utilizados por las personas para seleccionar colores (por ejemplo, de pinturas o tintas) de una rueda o paleta de colores. Este color o sistema está considerablemente más cerca que el sistema RGB de la forma en que los humanos experimentan y describen las sensaciones de

color. En la terminología de los artistas, el tono, la saturación y el valor se refieren aproximadamente al matiz, la sombra y el tono.

El espacio de color HSV se formula observando el cubo de color RGB a lo largo de su eje gris (el eje que une los vértices blanco y negro), lo que da como resultado la paleta de colores de forma hexagonal que se muestra en la figura 3.2(a). El tono se expresa como un ángulo alrededor de un hexágono de color, normalmente utilizando el eje rojo como eje de referencia (0°). El componente de valor se mide a lo largo del eje del cono figura 3.2(b).

El extremo *V* = 0 del eje es negro. El extremo *V* = 1 del eje es blanco, que se encuentra en el centro del hexágono a todo color en la figura 3.2(a). Por lo tanto, este eje representa todos los tonos de gris. La saturación (pureza del color) se mide como la distancia desde el eje *V*.

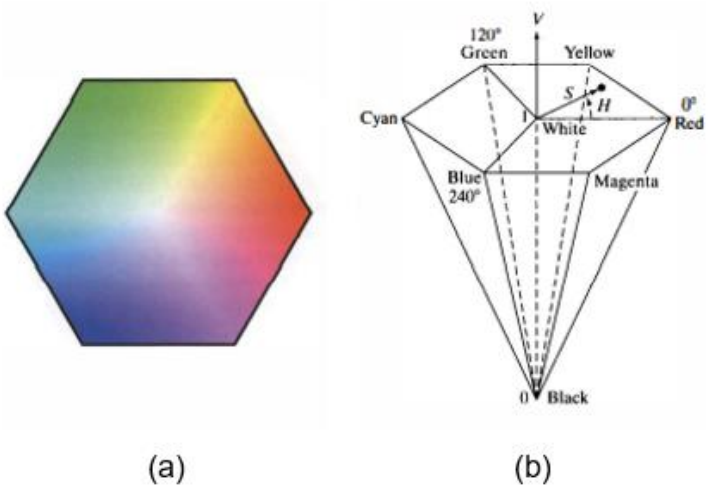


Figura 3.2 Representación del espacio de color HSV (González et al., 2009).

La conversión de color implementada en la librería de OpenCV para el espacio de color HSV[10] se muestra a continuación:

La transformación de color RGB a HSV está dada por:

$$V \leftarrow \max(R, G, B) \qquad \text{(Ec 3.1)}$$

$$S \leftarrow \begin{cases} \dfrac{V - \min(R, G, B)}{V} & if\ V \neq 0 \\ 0 & en\ otro\ caso \end{cases} \qquad \text{(Ec 3.2)}$$

10 https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv

$$H \leftarrow \begin{cases} \dfrac{60(G-B)}{V-\min(R,G,B)} & si\ V = R \\ 120 + \dfrac{60(B-R)}{V-\min(R,G,B)} & si\ V = G \\ 240 + 60(R-G) & si\ V = B \\ 0 & si\ R = G = B \end{cases} \qquad \text{(Ec 3.3)}$$

Si $H < 0$ entonces $H \leftarrow H + 360$. En salida $0 \leq V \leq 1{,}0 \leq S \leq 1{,}0 \leq H \leq 360$.

Los datos se convierten al tipo de dato de destino:

Imágenes de 8 bits: $V \leftarrow 255V, S \leftarrow 255S, H \leftarrow \frac{H}{2}(para\ ajustar\ de\ 0\ a\ 255)$.

Imágenes de 32 bits: *H*, *S*, *V* se dejan como están.

b) Modelo de Color YCrCb

De acuerdo con (González et al., 2009), el espacio de color YCbCr se usa ampliamente en video digital. En este formato, la información de luminancia está representada por un solo componente, *Y*, y la información de color se almacena como dos componentes de diferencia de color, *Cb* y *Cr*. El componente *Cb* es la diferencia entre el componente azul y un valor de referencia, y el componente *Cr* es la diferencia entre el componente rojo y un valor de referencia, ver figura 3.3.

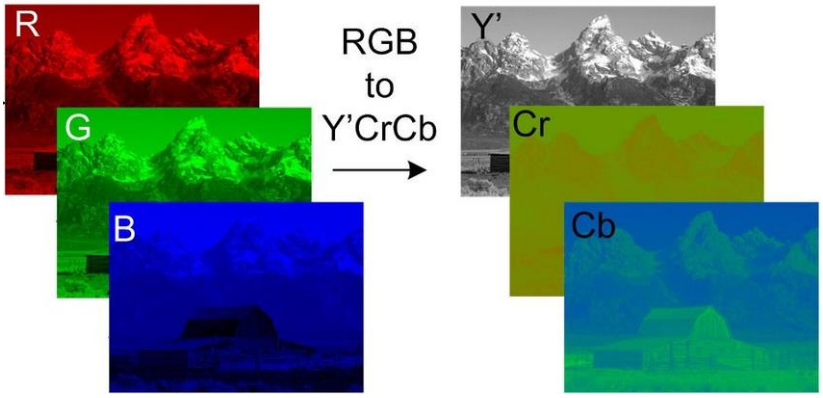


Figura 3.3 Representación de una imagen RGB convertida a YCrCb (Assi et al., 2016).

La conversión de color implementada en la librería de OpenCV para el espacio de color YCrCb[11] se muestra a continuación:

11 https://docs.opencv.org/3.4/de/d25/imgproc_color_conversions.html#color_convert_rgb_hsv

La transformación de color RGB a YCrCb está dada por:

$$Y \leftarrow 0.299 \cdot R + 0.587 \cdot G + 0.114 \cdot B \quad \text{(Ec 3.4)}$$

$$Cr \leftarrow (R - Y) \cdot 0.713 + delta \quad \text{(Ec 3.5)}$$

$$Cb \leftarrow (B - Y) \cdot 0.564 + delta \quad \text{(Ec 3.6)}$$

La transformación de color YCrCb a RGB está dada por:

$$R \leftarrow Y + 1.403 \cdot (Cr - delta) \quad \text{(Ec 3.7)}$$

$$G \leftarrow Y - 0.714 \cdot (Cr - delta) - 0.344 \cdot (Cb - delta) \quad \text{(Ec 3.8)}$$

$$B \leftarrow Y + 1.773 \cdot (Cb - delta) \quad \text{(Ec 3.9)}$$

Donde:

$$delta = \begin{cases} 128 & para\ imagenes\ de\ 8\ bits \\ 32768 & para\ imagenes\ de\ 16\ bits \\ 0.5 & para\ imagenes\ de\ tipo\ flotante \end{cases}$$

3.2.2 Ecualización del Histograma

Existen varias técnicas para mejorar las condiciones iniciales de las imágenes. En general, los conjuntos de datos revisados no tienen grandes cambios en la intensidad luminosa, por lo cual se considera que con una técnica sencilla como la ecualización del histograma puede mejorarse su calidad.

La ecualización de histograma redistribuye uniformemente los valores de nivel de gris de los píxeles dentro de una imagen de modo que el número de píxeles en cualquier nivel de gris sea aproximadamente el mismo. La transformación de ecualizar una imagen en nivel de gris es:

$$g_i = \frac{M - 1}{n_t} \sum_{j=0}^{i} n_i \quad \text{(Ec 3.10)}$$

donde n_t es el número total de píxeles de la imagen, n_i es el número de píxeles en el nivel de gris *i*, y *M* es el número total de niveles de gris posibles (Myler & Weeks, 1993).

3.3 Herramientas de Localización del Rostro

3.3.1 MediaPipe

MediaPipe Face Detection (MediaPipe, 2020) es una herramienta de detección de rostros que viene con 6 puntos de referencia y compatibilidad con múltiples rostros. Se basa en BlazeFace (Bazarevsky et al., 2019), un detector de rostros liviano y de buen rendimiento diseñado para la GPU móvil. El rendimiento en tiempo real del detector permite la estimación de puntos clave faciales en 3D (por ejemplo, MediaPipe Face Mesh que cuenta con 468 puntos faciales, como se puede observar en la figura 3.4).

También, gracias a que es una herramienta multiplataforma, el desarrollador puede configurar la aplicación creada con MediaPipe para administrar los recursos de manera eficiente (tanto para CPU como GPU) para lograr un rendimiento de baja latencia (Lugaresi et al., 2019).

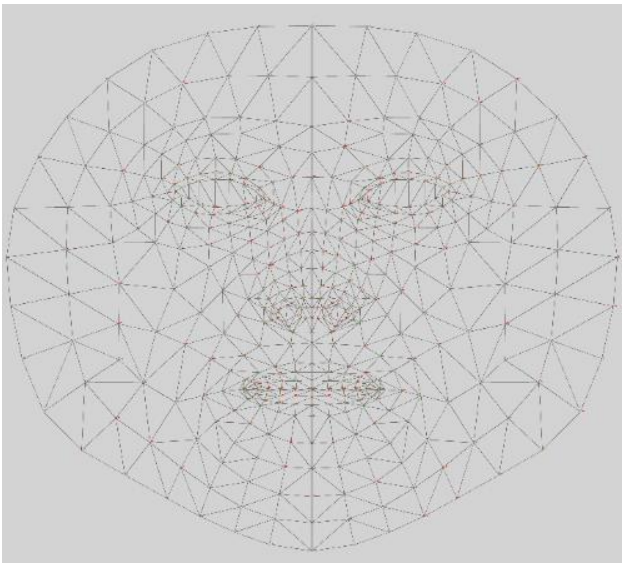


Figura 3.4 Plantilla de puntos faciales de MediaPipe [12].

3.3.2 Detector Facial DNN en OpenCV

DNN de OpenCV contiene el modelo Single Shot Multibox Detector (SSD) (Liu et al., 2016) y ResNet-10 (Anisimov & Khanova, 2017) como la arquitectura principal para trabajar en un enfoque de tiempo real en la detección del rostro (Nagrath et al., 2021).

12https://github.com/google/mediapipe/blob/master/mediapipe/modules/face_geometry/data/canonical_face_model_uv_visualization.png

DNN requiere de dos archivos de *Caffe*, los cuales son *deploy.prototxt* que define la arquitectura de la red y *res10_300x300_ssd_iter_140000.caffemodel* contiene los pesos de las capas (Kavitha et al., 2021), estos archivos se pueden descargar desde el repositorio de *Github*[13]

3.3.3 MTCNN

Combinan cascaded CNNs by multi-task learning (K. Zhang et al., 2016b), implementan una estructura multitarea en cascada profunda haciendo uso de la correlación inherente entre ellas para tener un aumento en el rendimiento. Cada una de las capas en cascada consta de tres etapas de redes convolucionales profundas, cuidadosamente diseñadas, prediciendo los puntos de referencia del rostro de una forma gruesa a una forma fina. Además de detectar la ubicación del rostro, realiza la alineación de las partes importantes del rostro, etapa necesaria sobre todo en ambientes reales.

3.3.4 DLIB

Dlib (Dlib, 2022) es un conjunto de herramientas modernas que contiene algoritmos de aprendizaje automático para crear software complejo en el lenguaje C++, para resolver problemas del mundo real. Se utiliza tanto en la industria como en el mundo académico en una amplia gama de dominios y se puede usar en diversas plataformas, entre ellas en Python.

Dlib se integra del detector de rostros HOG + Linear SVM que es rápido y eficiente. Pero, debido a la naturaleza de cómo funciona el descriptor Histogram of Oriented Gradients (HOG), no es invariable a los cambios en la rotación y el ángulo de visión (Rosebrock, 2021).

Adicionalmente, la biblioteca Dlib cuenta con un detector de referencias faciales pre-entrenado, que se utiliza para estimar la ubicación de 68 coordenadas (x, y) que se asignan a estructuras o elementos faciales, como se puede ver en la figura 3.5 (Rosebrock, 2017).

13 https://github.com/alvareson/caffe_model_for_dace_detection

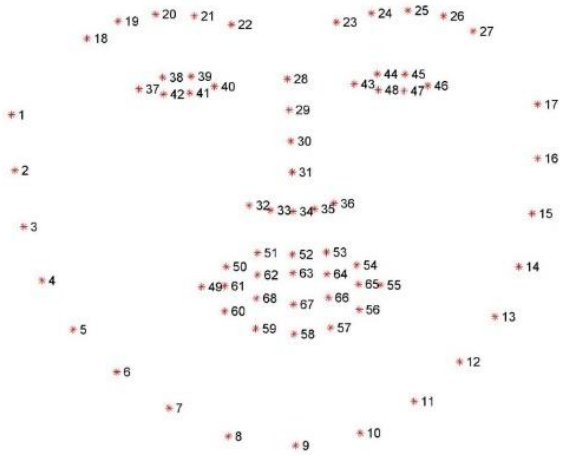


Figura 3.5 Plantilla de puntos faciales de Dlib (Rosebrock, 2017).

3.4 Descriptores de Textura

3.4.1 Local Binary Patterns (LBP)

El primer paso para construir el descriptor de textura patrones binarios locales (LBP) (Rosebrock, 2015) es convertir la imagen a escala de grises. Para cada píxel de la imagen en escala de grises, se selecciona una vecindad de tamaño *r* que rodea al píxel central. Luego se calcula un valor LBP para este píxel central y se almacena en la matriz *2D* de salida con el mismo ancho y alto que la imagen de entrada. En la figura 3.6 se aprecia el LBP original en un vecindario de píxeles fijo de *3 x 3*.

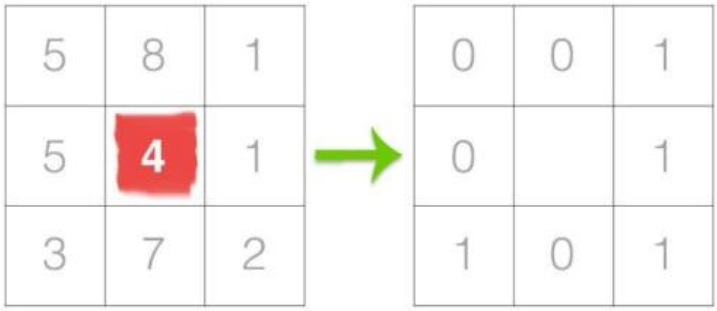


Figura 3.6 Ejemplo de LBP en una vecindad de 8 píxeles, con un umbral para construir un conjunto de 8 dígitos binarios (Rosebrock, 2015).

En la figura anterior, se toma el píxel central y se compara con su vecindario de *8* píxeles, si la intensidad del píxel central es mayor o igual que su vecino, se establece un valor de 1, en caso contrario se establece un valor de 0.

A partir de ahí, se calcula el valor de LBP para el píxel central. Es posible comenzar desde cualquier píxel vecino y avanzar en sentido horario o antihorario, pero el orden debe ser constante para todos los píxeles de la imagen y de todas las imágenes del conjunto de datos. Los resultados de esta prueba binaria se almacenan en una matriz de *8* bits, que luego se convierten a decimal, como se aprecia en la figura 3.7.

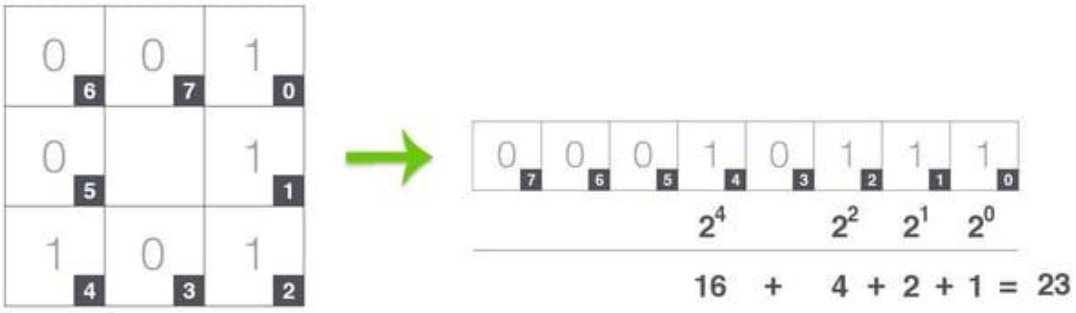


Figura 3.7 Conversión de la vecindad 8 a decimal (Rosebrock, 2015).

En el ejemplo de la figura 3.7, se comienza vecino superior derecho y se avanza en el sentido de las agujas del reloj acumulando la cadena binaria a medida que se avanza; posteriormente, se convierte la cadena binaria a decimal, dando un valor de *23*, dicho valor se almacena en la matriz LBP *2D* de salida como se precia en la figura 3.8.

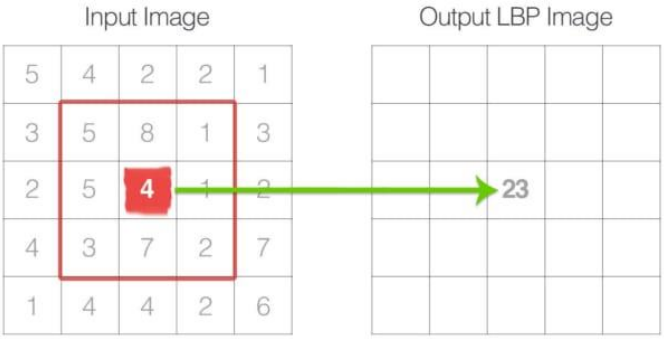


Figura 3.8 Representación del valor LBP calculado y almacenado en la matriz de salida (Rosebrock, 2015).

Este proceso de creación de umbrales, acumulación de cadenas binarias y almacenamiento del valor decimal de salida en la matriz LBP se repite para cada píxel de la imagen de entrada. El último paso es calcular un histograma sobre la matriz LBP de salida. La matriz LBP *2D* tiene un valor mínimo de 0 y un valor máximo de *255*, lo que permite construir un histograma de *256* valores LBP como vector de características final.

3.4.2 Binarized Statistical Image Features (BSIF)

El descriptor BSIF (Benzaoui et al., 2017) o funciones de imágenes estadísticas binarizadas fue propuesto por Kannala y Rahtu (Kannala & Rahtu, 2012) y se ha utilizado para el reconocimiento facial y la clasificación de texturas. La técnica se basa en las metodologías Local Binary Patterns (LBP) y Local Phase Quantization (LPQ)y la idea detrás de BSIF consiste en aprender automáticamente un conjunto fijo de filtros a partir de un pequeño conjunto de imágenes naturales, en lugar de usar filtros hechos a mano como LBP o LPQ. BSIF implica un aprendizaje automático en lugar de un ajuste manual para obtener una representación estadísticamente significativa de la imagen, lo que permite la codificación de la información efectiva mediante el uso de una simple cuantificación por elementos. Este aprendizaje también proporciona una manera fácil y flexible de ajustar la longitud del descriptor y adaptarlo a las aplicaciones que presentan imágenes que contienen características difíciles e inusuales.

Para caracterizar las propiedades de la textura en cada subregión de la imagen, se utilizan luego los histogramas de las etiquetas BSIF. El valor de cada elemento (bit) en la cadena binaria del código BSIF se calcula mediante la respuesta de binarización de un filtro lineal, con un umbral en cero. Cada bit está asociado a un filtro diferente y la cadena de bits determina el número de filtros utilizados. El conjunto de filtros se aprende a partir de un conjunto de parches de imágenes naturales al maximizar la independencia estadística de las respuestas para cada filtro.

En la práctica, dado un parche de una imagen *X* de tamaño lxl píxeles y un filtro lineal w_i del mismo tamaño, la respuesta del filtro s_i se obtiene por:

$$s_i = \sum_{u,v} w_i(u,v)X(u,v) = w_i^T x \qquad \text{(Ec 3.11)}$$

donde los vectores *w* y *x* contienen los píxeles de w_i y x

La característica binarizada b_i se obtiene por el conjunto de: $b_i = 1$ si $s_i > 0$ y $b_i = 0$ en caso contrario. Los filtros w_i se aprenden utilizando independent component analysis (ICA) (Comon, 1994) que maximiza la independencia estadística de s_i

El descriptor BSIF tiene dos parámetros: el tamaño del filtro l y la longitud de la cadena binaria n. Los filtros originales propuestos por Kannala y Rahtu, que están disponibles en línea para su uso y prueba, se aprendieron a partir de 50,000 parches de imágenes. La figura 3.9 muestra un ejemplo de filtros obtenidos con $l = 7$ y $n = 8$.

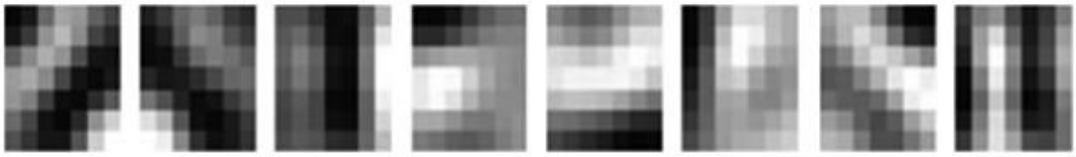


Figura 3.9 Un banco de filtros BSIF 7×7 aprendidos de imágenes naturales (Benzaoui et al., 2017).

Entre las limitaciones observadas del operador BSIF se tienen que los filtros originales propuestos por Kannala y Rahtu se aprendieron a partir de un conjunto fijo de parches de imágenes texturizadas, aleatorias y naturales. Los filtros se pueden cambiar dependiendo de las imágenes a analizarse.

3.5 Clasificadores

El campo del aprendizaje automático ha recibido varias definiciones formales en la literatura. Arthur Samuel, definió el aprendizaje automático como “un campo de estudio que brinda a las computadoras la capacidad de aprender sin ser programadas explícitamente” (Samuel, 1959). Utilizando un léxico de ciencias de la computación, Tom Mitchell lo presentó como “un programa de computadora aprende de la experiencia (E) con respecto a alguna clase de tareas (T) y medida de desempeño (P), si su desempeño en tareas en T, como medido por P, mejora con la experiencia E ” (Mitchell & others, 1997). Ethem Alpaydin en su libro de texto definió el aprendizaje automático como el campo de “Programar computadoras para optimizar un criterio de desempeño usando datos de ejemplo o experiencias pasadas” (Alpaydin, 2014). Estas diversas definiciones comparten la noción de entrenar a las computadoras para que realicen tareas de manera inteligente más allá del procesamiento numérico tradicional al aprender el entorno circundante a través de ejemplos repetidos (Naqa & Murphy, 2015).

Sin embargo, para este trabajo se consideraron dos clasificadores tradicionales como se mencionan a continuación

3.5.1 Random Forest

También conocido como clasificador de bosque aleatorio (Pedregosa et al., 2011a) es un meta estimador que crea una serie de clasificadores de árboles de decisión a partir de submuestras del conjunto de datos y utiliza el promedio para mejorar la precisión predictiva y controlar el sobreajuste.

En los bosques aleatorios cada árbol del conjunto se construye a partir de una muestra extraída con reemplazo del conjunto de entrenamiento. Además, al dividir

cada nodo durante la construcción de un árbol, la mejor división se encuentra entre todas las entidades de entrada o un subconjunto aleatorio de tamaño max_features.

El propósito de estas dos fuentes de aleatoriedad es disminuir la varianza del estimador forestal. De hecho, los árboles de decisión individuales suelen exhibir una gran variación y tienden a sobre ajustarse. La aleatoriedad inyectada en los bosques produce árboles de decisión con errores de predicción algo desacoplados. Al tomar un promedio de esas predicciones, algunos errores pueden cancelarse. Los bosques aleatorios logran una varianza reducida al combinar árboles diversos, a veces a costa de un ligero aumento en el sesgo. En la práctica, la reducción de la varianza suele ser significativa, por lo que genera un mejor modelo general.

La implementación de scikit-learn combina clasificadores promediando su predicción probabilística, en lugar de permitir que cada clasificador vote por una sola clase. Siendo la configuración predefinida la que se muestra a continuación:

*Class sklearn.ensemble.RandomForestClassifier(n_estimators=100, *, criterion='gini', max_depth=None, min_samples_split=2, min_samples_leaf=1, min_weight_fraction_leaf=0.0, max_features='sqrt', max_leaf_nodes=None, min_impurity_decrease=0.0, bootstrap=True, oob_score=False, n_jobs=None, random_state=None, verbose=0, warm_start=False, class_weight=None, ccp_alpha=0.0, max_samples=None)*

3.5.2 Máquinas de Vector Soporte (MVS)

El clasificador Máquinas de Vector Soporte (SVM por Support Vector Machines) (Parker, 2010) tiene como objetivo encontrar el hiperplano de margen máximo. Una línea divide los datos bidimensionales en dos partes; un plano divide los datos tridimensionales en dos partes; y un hiperplano es una función lineal que divide datos N-dimensionales en dos partes. El hiperplano de margen máximo siempre está lo más lejos posible de ambos conjuntos de datos. Las máquinas de vectores de soporte también pueden encontrar límites no lineales entre clases, que es su otra gran ventaja sobre otros métodos. Esto no se logra encontrando trayectorias lineales curvas o por partes entre los vectores de características de cada clase, sino que, lo logra transformando esos vectores de características para que se pueda encontrar un límite lineal.

Los datos se han proyectado en un espacio de características de mayor dimensión. Esta transformación utiliza un kernel, que es la función que proyecta los datos. Hay muchos núcleos posibles, la figura 3.10 muestra el resultado de usar una Gaussiana (una función de base radial), pero se pueden usar polinomios y otras funciones, dependiendo de los datos.

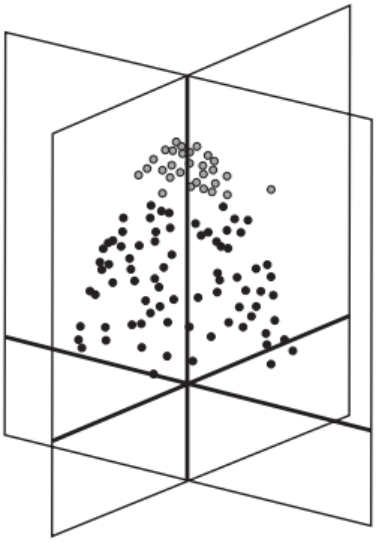


Figura 3.10 Vectores de características proyectadas utilizando una función de base radial para separarlos en un plano (Parker, 2010).

Las ventajas de las máquinas de vectores de soporte son: (Pedregosa et al., 2011b).

- Eficaz en espacios de altas dimensiones.
- Todavía efectivo en casos donde el número de dimensiones es mayor que el número de muestras.
- Utiliza un subconjunto de puntos de entrenamiento en la función de decisión (llamados vectores de soporte), por lo que también es eficiente en memoria.
- Versátil: se pueden especificar diferentes kernels para la función de decisión. Generalmente, se proporcionan kernels comunes, pero también es posible especificar o personalizarlo.

Las desventajas de las máquinas de vectores de soporte (Pedregosa et al., 2011b).

- Cuando el número de funciones es mayor que el número de muestra; es crucial evitar el ajuste excesivo al elegir las funciones del núcleo y el término de regularización.
- Las SVM no proporcionan directamente estimaciones de probabilidad, estas se calculan mediante una costosa validación cruzada de cinco veces.

La configuración pre definida se muestra a continuación (Pedregosa et al., 2011b):

Class sklearn.svm.SVC (, C=1.0, kernel='rbf', degree=3, gamma='scale', coef0=0.0, shrinking=True, probability=False, tol=0.001, cache_size=200, class_weight=None, verbose=False, max_iter=1, decision_function_shape='ovr', break_ties=False, random_state=None)*

3.6 Métricas de Evaluación

La matriz de confusión es una herramienta que permite visualizar el desempeño de un algoritmo de aprendizaje supervisado. Cada columna de la matriz representa el número de predicciones de cada clase, mientras que cada fila representa a las instancias en la clase real, ver figura 3.11. Es decir, permite ver qué tipos de aciertos y errores está teniendo el modelo a la hora de pasar por el proceso de aprendizaje con los datos, (J. I. Barrios, 2019).

Figura 3.11 Matriz de confusión (J. I. Barrios, 2019).

Estas 4 opciones integran a la matriz de confusión.

Verdadero positivo: El valor real es positivo y la prueba predijo también que era positivo.

Verdadero negativo: El valor real es negativo y la prueba predijo también que el resultado era negativo.

Falso positivo: El valor real es negativo, y la prueba predijo que el resultado es positivo. Esto es lo que en estadística se conoce como error tipo I.

Falso negativo: El valor real es positivo, y la prueba predijo que el resultado es negativo. Esto es lo que en estadística se conoce como error tipo II

A partir de estos valores se calculan las siguientes métricas (J. I. Barrios, 2019):

Exactitud

La exactitud (en inglés, “Accuracy”) se refiere a lo cerca que está el resultado de una medición del valor verdadero. En términos estadísticos, la exactitud está relacionada con el sesgo de una estimación.

$$Exactitud = \frac{Vp + Vn}{(Vp + Fp + Fn + Vn)} \quad \text{(Ec 3.12)}$$

Precisión

La Precisión (en inglés “Precision”) Se refiere a la dispersión del conjunto de valores obtenidos a partir de mediciones repetidas de una magnitud. Cuanto menor es la dispersión mayor la precisión.

$$Precisión = \frac{Vp}{(Vp + Fp)} \quad \text{(Ec 3.13)}$$

Sensibilidad

La Sensibilidad (“Recall” o “Sensitivity”), También se conoce como Tasa de Verdaderos Positivos (True Positive Rate) ó TP. Es la proporción de casos positivos que fueron correctamente identificadas por el algoritmo.

$$Sensibilidad = \frac{Vp}{(Vp + Fn)} \quad \text{(Ec 3.14)}$$

F1 Score

Métrica muy empleada ya que resume la precisión y sensibilidad en una sola métrica. Por ello es de gran utilidad cuando la distribución de las clases es desigual.

$$f1\ score = 2 * \frac{Sensibilidad * Precisión}{(Sensibilidad + Precisión)} \quad \text{(Ec 3. 15)}$$

Capítulo 4

Experimentación y Resultados

4.1 Datasets

A continuación, se describen los *datasets* ocupados en la experimentación realizada.

MaskedFace-Net (Cabani et al., 2021): El *dataset* cuenta con más de 137,016 imágenes de rostros enmascarados organizados en dos subconjuntos:

- El conjunto de rostros correctamente enmascarados (cubriendo la nariz y boca) (denominado CMFD por sus siglas en inglés), que integra el 49% de las imágenes.
- El conjunto imágenes de rostros incorrectamente enmascarados (denominado IMFD por sus siglas en inglés) corresponde al 51% de las imágenes.

Este *dataset* se construyó agregando un cubrebocas simulado usando el *dataset* FFHQ[14] por su gran variedad en términos de edad, etnia, punto de vista, iluminación y fondo de imagen. Ejemplos de las imágenes se pueden ver en la figura 4.1:

Figura 4.1 Muestra de imágenes de los dos subconjuntos de MaskedFace-Net (Cabani et al., 2021).

14 https://github.com/NVlabs/ffhq-dataset

RMFRD (Wang et al., 2020): Reconocimiento facial enmascarado del mundo real *dataset* (RMFRD por sus siglas en inglés). Incluye 5,000 fotografías de 525 personas con máscaras y 90,000 imágenes de los mismos 525 sujetos sin máscaras. Se menciona que las imágenes contienen el rostro de vista frontal de figuras públicas, bajadas de recursos masivos de internet, por lo cual, las imágenes presentan alta variabilidad. En la figura 4.2 se pueden apreciar ejemplo de las imágenes del *dataset* RMFRD.

Figura 4.2 Ejemplo de imágenes del dataset RMFRD (Wang et al., 2020).

Flickr-Faces-HQ (FFHQ) [15]:

El conjunto de datos consta de 70,000 imágenes PNG de alta calidad con una resolución de *128×128* y contiene una variación considerable en términos de edad, origen étnico y fondo de la imagen. También tiene una buena cobertura de accesorios como anteojos, gafas de sol, sombreros, etc.

Las imágenes fueron rastreadas desde Flickr, heredando así todos los sesgos de ese sitio web, y automáticamente alineadas y recortadas usando Dlib. Se usaron varios filtros automáticos para podar el conjunto y, finalmente, se usó Amazon Mechanical Turk para eliminar las estatuas, pinturas o fotos de fotos ocasionales. En la figura 4.3 se pueden apreciar ejemplo de las imágenes del *dataset*.

15 https://github.com/NVlabs/ffhq-dataset

Figura 4.3 Muestra de imágenes de FFHQ[16].

Conjunto Propio: Consta de 583 imágenes (de 3 personas) adquiridas en diferentes días lo que provoca cambios de iluminación, rotación del rostro y diferentes tipos de oclusión, además de tener mayor variabilidad en el tipo del cubrebocas y en condiciones reales, como se puede observar en la figura 4.4. Las imágenes tienen un tamaño de *2740x2740* píxeles, en formato RGB.

Figura 4.4 Muestra de imágenes dataset propio.

De los *datasets* anteriormente presentados se tomaron muestras, creando un conjunto nuevo con un total de 3,133 imágenes, las cuales se ocuparon para realizar la extracción de características, información con la cual se entrenaron los clasificadores.

16 https://github.com/NVlabs/ffhq-dataset

4.2 Casos de Experimentación

La sección está integrada por cuatro casos de prueba.

Caso 1: Comparación de cuatro herramientas de localización del rostro en una imagen. El objetivo de esta prueba fue comparar el desempeño de cuatro sistemas de detección de rostros en entornos no controlados. Es decir, con diferente escala, rotación, intensidad luminosa y, sobre todo, evaluar su capacidad de tratar la oclusión causada por prendas o complementos del vestuario como pueden ser sombreros, gorras, cascos, lentes graduados, cabello, incluso maquillaje, diversos tonos de piel y, sin olvidar la oclusión del rostro ocasionada por el uso del cubrebocas. Para la evaluación se consideran los modelos pre entrenados para localizar el rostro.

Caso 2: Evaluación de dos descriptores de textura. El objetivo de esta prueba fue evaluar el desempeño del descriptor clásico de textura LBP con respecto al descriptor BSIFT. En este caso, se consideraron tres formas de llevar a cabo la normalización de los vectores resultados.

Caso 3: Evaluación de la imagen sin y con procesamiento. La realización de esta evaluación se lleva a cabo en dos etapas. El objetivo de la primera parte es evaluar el sistema desarrollado teniendo como entrada la imagen original en formato RGB. El objetivo de la segunda parte de la prueba es evaluar el sistema al aplicar un procesamiento a la imagen para tratar de minimizar los efectos de los cambios de iluminación, mediante la transformación de la imagen RGB a YCbCr y la ecualización del canal “Y”. En ambas pruebas se utilizan los algoritmos de clasificación Random Forest y Máquinas de Vector Soporte.

Caso 4: Evaluación del tiempo. El objetivo de esta prueba fue evaluar el tiempo promedio de respuesta de cada imagen analizada. En este caso se utilizó un video de 2 min 44 segundos que equivale a 1,756 fotogramas.

En las siguientes secciones se detallan los casos mencionados.

4.2.1 Caso 1: Comparación de Cuatro Herramientas de Localización del Rostro en una Imagen

Existen varias propuestas para la detección del rostro en una imagen, capaces de localizar todos los rostros presentes simultáneamente; sin embargo, se eligió Dlib, MTCNN, Detector Facial DNN en OpenCV y MediaPipe ya que presentan modelos

pre entrenados con lo cual no requieren un entrenamiento adicional y las hace fáciles de implementar.

Para llevar a cabo la evaluación de la robustez de las herramientas considerando variaciones reales del ambiente, se tomó una muestra de 130 imágenes de cada uno de los cuatro conjuntos públicos antes mencionados, analizándose un total de 520 fotogramas. Estas imágenes presentan rostros cubiertos con cubrebocas, fondos complejos, diferentes tonos de piel, escala, perspectiva, edad, etnia y la presencia de uno o más rostros en la imagen. Los rostros pueden también contener accesorios como lentes, cabello, gorras, bufandas, además de tener aproximadamente la mitad del mismo ocluido por el uso del cubrebocas de distintos colores.

Dado que las imágenes consideradas de los distintos *datasets* tenían una dimensionalidad diferente entre sí, fue necesario llevar a cabo un redimensionamiento (normalización) de las mismas a 300x300 píxeles, esto ayuda también a su visualización.

También se renombraron las imágenes para tener uniformidad en el nombre y que el sistema pueda leerlas y cargarlas de manera automática. No se llevó a cabo otro preprocesamiento de las mismas, ya que el objetivo era comprobar la robustez de cada herramienta con los parámetros originales al realizar la instalación de las mismas.

En la tabla 4.1 se pueden observar los resultados obtenidos por cada herramienta, mediante las métricas de exactitud, precisión, sensibilidad y F1 score.

Tabla 4.1 Resultados obtenidos de las herramientas.

Métrica	**Herramienta**			
	Dlib	**DNN**	**MediaPipe**	**MTCNN**
Exactitud	0.530	0.559	**0.867**	0.373
Precisión	**1.00**	0.564	0.975	**1.00**
Sensibilidad	0.530	**0.984**	**0.886**	0.373
F1 Score	0.693	0.718	**0.929**	0.543

Como se puede observar en la tabla 4.1, Dlib y MTCNN tienen buena precisión gracias a que localiza perfectamente todos los rostros frontales presentes en la imagen. La detección correcta la realizan aún ante la presencia del uso de cubrebocas, el uso de casco, presencia de cabello en la frente, uso de lentes graduados y uso de sombreros. Realmente, el factor principal que limita su desempeño es que ambas herramientas necesitan poder localizar ambos ojos de

manera completa, por ello, obtienen un valor bajo en las métricas de exactitud, sensibilidad y F1 Score.

Por su parte, DNN y MediaPipe son las herramientas más eficientes que generan una buena respuesta ante las condiciones ambientales antes mencionadas, para la detección del rostro con oclusión obtienen 0.984% y 0.886% de sensibilidad respectivamente. La herramienta MediaPipe, tiene el mejor resultado de 0.867% de exactitud sobre las otras tres herramientas al detectar el rostro, aun cuando existe presencia de rotaciones que limitan la presencia completa de las partes del rostro.

En la figura 4.5 se pueden observar dos ejemplos de la detección realizada con las 4 herramientas analizadas. Para cada ejemplo, la imagen de la parte superior izquierda se obtuvo utilizando el detector facial Dlib, la imagen superior derecha se obtuvo con detector facial DNN, la imagen inferior izquierda se obtuvo con el detector MediaPipe y finalmente la imagen inferior derecha se obtuvo con el detector MTCNN. Dlib a pesar de tener una buena precisión no es invariante a rotaciones y tiene problemas con la localización del rostro que presenta una gran oclusión.

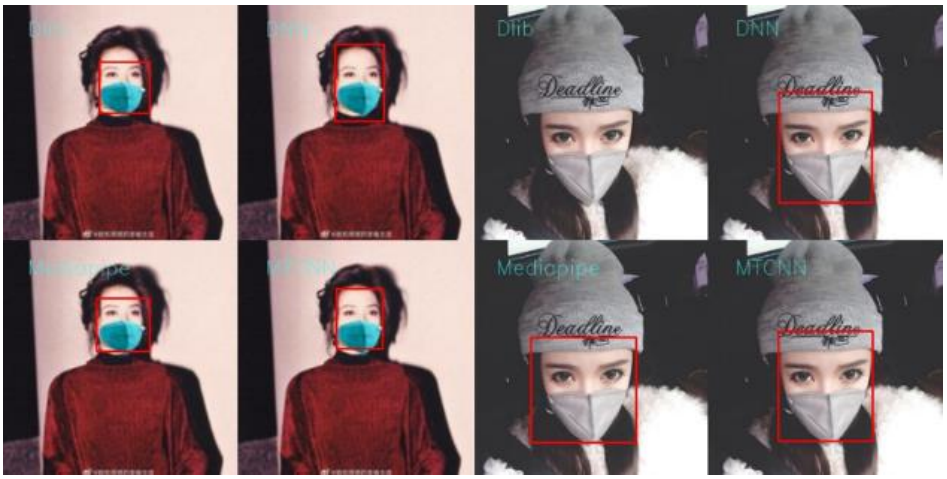


Figura 4.5 Ejemplo de los resultados obtenidos con cada detector facial.

Las cuatro herramientas trabajan bien ante cambios en la escala, localizando los rostros presentes en regiones pequeñas y en regiones de tamaño medio (considerando la dimensión de toda la imagen). Sin embargo, MediaPipe suele tener problemas con rostros pequeños (más lejanos en la imagen) pero detecta mejor los rostros con una mayor apreciación de rotación. Por otra parte, la herramienta DNN genera falsos positivos si el rostro cubre casi por completo toda la región correspondiente a la imagen a analizar; es decir, localiza perfectamente el rostro en la imagen, pero también genera un rectángulo en la parte inferior derecha de la imagen, es decir, presenta falsos positivos, como se observa en la figura 4.6.

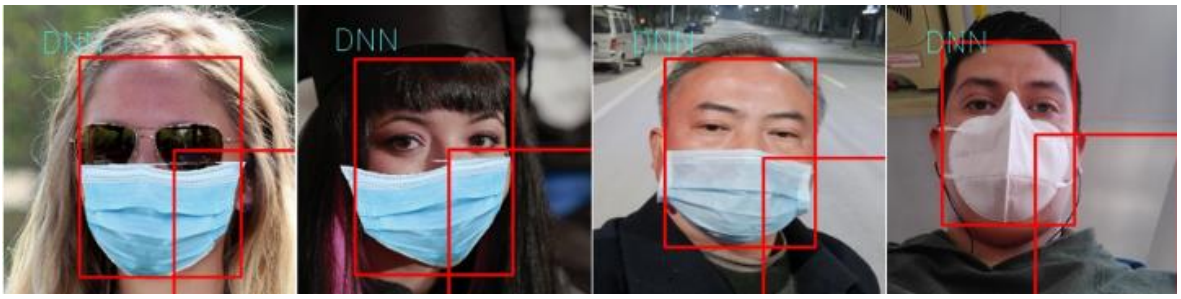


Figura 4.6 Visualización de Falsos positivos con DNN.

Concluyendo, MediaPipe es una herramienta relativamente nueva con la suficiente robustez para realizar la tarea de localización bajo los supuestos mencionados, obteniendo una precisión del 97.5%, una exactitud del 86.7% y un 92.9 % de F1 score para este trabajo.

4.2.2 Caso 2: Evaluación de Dos Descriptores de Textura

- **Caso A:** Técnicas de descripción de textura en el entrenamiento

De manera aleatoria se tomaron 1,000 imágenes, 500 imágenes de rostros libres del cubrebocas y 500 imágenes con cubrebocas de los 4 *datasets* considerados. El conjunto de entrenamiento se integra de 1,600 instancias. Como ya se mencionó se consideraron las técnicas LBP y BSIF cuyas descripciones se normalizaron en 3 formas. Las primeras dos fueron utilizando la normalización proporcionada por OpenCV que toma en cuenta los rangos de *0-1* y *0-255*. Adicionalmente, se agregó otra forma que consiste en la división de cada valor del vector entre la suma total de todos los valores del histograma (por columna), más un valor de *1e-7* para evitar que se genere un error de división. Obteniendo los resultados que se muestran en la tabla 4.2 con la métrica accuracy.

Tabla 4.2 Resultados del entrenamiento.

Textura	**Métrica**	**Clasificador**	
		RF	**SVM**
TRAIN 0-1			
BSIF	Accuracy	**0.882**	0.861291
LBP	Accuracy	**0.876**	0.858287
TRAIN 0-255			
BSIF	Accuracy	0.898	**0.923885**
LBP	Accuracy	**0.918**	0.914872
TRAIN 1e-7			
BSIF	Accuracy	**0.882**	0.858287
LBP	Accuracy	**0.882**	0.858287

Como se puede observar, BSIF fue ligeramente superior con la normalización de *0-1* en ambos clasificadores. El resultado obtenido con la normalización de *0-255* mantiene la ligera superioridad de BSIF con SVM, mientras que LBP obtiene su mejor resultado con Random Forest. Y finalmente, con la normalización identificada como *1e-7*, se obtiene el mismo desempeño con ambas técnicas.

- **Caso B:** Técnicas de descripción de textura con el conjunto de prueba

El conjunto de prueba se integra de 400 instancias diferentes a las consideradas en la fase de entrenamiento. Para la evaluación se realizaron pruebas considerando las tres diferentes formas de normalización. En este caso se consideraron 4 métricas, como se aprecia en las tablas 4.3, 4.4 y 4.5.

Tabla 4.3 Resultados con el conjunto de prueba y la normalización de 0-1.

Métrica	**TEST BSIF 0-1**			**TEST LBP 0-1**	
	Clasificador			**Clasificador**	
	RF	SVM		RF	SVM
ACCURACY	0.727272	**0.752066**		**0.809917**	0.752066
RECALL	0.652777	**0.654320**		**0.705128**	0.654320
F1 SCORE	0.740157	**0.779411**		**0.827067**	0.779411
PRECISION	0.854545	**0.963636**		**1.0**	0.963636

Como puede observarse en la tabla 4.3, los resultados obtenidos con el conjunto de prueba muestran un ligero descenso en el desempeño. En este caso, ambos descriptores obtienen los mismos resultados con el clasificador MVS. Pero, la combinación de LBP con Random Forest logra el mejor desempeño, incluso alcanzando el 100% de precisión.

Aplicando la normalización de *0-255*, ver tabla 4.4, LBP obtiene los mejores resultados y supera los resultados de BSIF en ambos clasificadores, alcanzando con Random Forest una exactitud del 90%. La técnica de BSIF obtiene una buena precisión en combinación con SVM, pero el resto de las métricas muestran un rendimiento bajo.

Tabla 4.4 Resultados de evaluación con normalización de 0-255.

Métrica	**TEST BSIF 0-255**			**TEST LBP 0-255**	
	Clasificador			**Clasificador**	
	RF	SVM		RF	SVM
ACCURACY	0.644628	**0.776859**		**0.900826**	0.884297
RECALL	0.6	**0.675**		**0.841269**	0.825396
F1 SCORE	0.626086	**0.8**		**0.898305**	0.881255
PRECISION	0.654545	**0.981818**		**0.963636**	0.945454

Tabla 4.5 Resultados de evaluación con normalización de 1e-7.

Métrica	TEST BSIF 1e-7		TEST LBP 1e-7	
	Clasificador		Clasificador	
	RF	SVM	RF	SVM
ACCURACY	**0.752066**	**0.752066**	**0.760330**	0.752066
RECALL	**0.662337**	0.654320	**0.675675**	0.654320
F1 SCORE	0.772727	**0.779411**	0.775193	**0.779411**
PRECISION	0.927272	**0.963636**	0.909090	**0.963636**

Finalmente, en la tabla 4.5 se pueden observar los resultados correspondientes con la normalización *1e-7*. En este caso, ambos descriptores tienen un desempeño bajo comparado con el obtenido en el entrenamiento, incluso con el logrado con la normalización *0-255*, con un máximo de exactitud del 76% con la combinación de LBP y Random Forest.

Analizando los resultados obtenidos con los datos de prueba, el rendimiento de los clasificadores baja considerablemente con las tres formas de normalización. Note que LBP es constante en los resultados con los datos de entrenamiento y prueba, obteniendo un mejor valor de accuracy ligeramente arriba de 90% en combinación con Random Forest (en la normalización de *0-255*) y SVM obtiene un accuracy apenas arriba del 88%.

4.2.3 Caso 3: Evaluación de la Imagen Sin y Con Preprocesamiento

Para llevar a cabo este experimento se tomaron muestras de los tres conjuntos de imágenes públicas y del conjunto creado propio (sección 4.1) formando uno nuevo con un total de 3,133 imágenes. Las imágenes presentan rostros cubiertos con cubrebocas, fondos complejos, diferentes tonos de piel, escala, edad, etnia y la presencia de uno o más rostros en la imagen. Los rostros pueden también contener accesorios como lentes, cabello, gorras, bufandas, además de tener aproximadamente la mitad del rostro ocluido por el uso del cubrebocas de distintos colores.

Dado que las imágenes de los distintos *datasets* tienen una dimensionalidad diferente entre sí, fue necesario llevar a cabo un redimensionamiento (normalización) de las mismas a *128x128* píxeles. También se renombraron las imágenes para tener uniformidad en el nombre y que el sistema pueda leerlas y procesarlas manera automática.

Para las pruebas, primero se consideró todo el *dataset* en un porcentaje de *80/20*, lo cual corresponde a 5,012 instancias para el entrenamiento y 1,254 elementos para las pruebas respectivamente. Posteriormente, se llevó a cabo el entrenamiento considerando la validación cruzada con *5 divisiones* y *10 divisiones*. Los clasificadores considerados son Random Forest y Support Vector Machines con el kernel *polinomial* y el kernel *rbf*.

Como resultado de la experimentación realizada en el caso 3, se considera el descriptor LBP (con la normalización de los datos de *0-255*). Es importante recordar que, la prueba sin procesamiento significa tomar la imagen en formato RGB como entrada y para la fase con procesamiento, la imagen RGB se transforma al espacio de color YCbCr, se ecualiza la banda *Y*, y la imagen resultante se vuelve a transformar a RGB.

- **Caso A:** Resultados de la Clasificación Sin Preprocesamiento

En la tabla 4.6 se pueden observar los resultados obtenidos de la clasificación *80/20*, con las métricas accuracy y F1 score. El clasificador Random Forest obtiene muy buenos resultados en el entrenamiento, logrando un 100% en ambas métricas. En la etapa de prueba, baja su desempeño alcanzando sólo el 84% de accuracy.

Por su parte, SVM tiene una mejor respuesta con el kernel *rbf* pero no logra superar a Random Forest al obtener un accuracy de 80%, en el entrenamiento y un 81% en la fase de validación. Los resultados obtenidos con el kernel Polinomial son ligeramente menores.

El tiempo de entrenamiento y validación también fueron considerados en este caso, logrando observar que Random Forest se entrena en *1* segundo, a diferencia de MVS que supera los *2* segundos en el entrenamiento (con ambos kernels). Se hace notar que MVS con el kernel polinomial muestra una disminución de su tiempo en la fase de prueba, sin lograr superar a Random Forest que obtiene un tiempo de 79 ms en el *test.*

Tabla 4.6 Resultado de la clasificación considerando el porcentaje de los datos a 80/20.

Imagen RGB					
80 / 20					
Random Forest					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.09988093	**1**	**1**	0.07919812	**0.84287012**	**0.86515978**
SVM-POLY					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
2.06978798	0.79155313	0.83723404	0.25371861	0.80108992	0.83956044
SVM-RBF					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
2.33734751	0.80517711	0.8454611	1.01486993	**0.81017257**	**0.84552846**

En la tabla 4.7 se pueden observar los resultados obtenidos con la validación cruzada de *5 divisiones*. En este caso, nuevamente el clasificador también Random Forest presenta el mejor desempeño de entrenamiento con 100% de accuracy y F1 score. Sin embargo, en la validación baja su desempeño y solo alcanza un 83% de accuracy y 91% de F1 score.

Por su parte, MSV con ambos kernels tiene un desempeño menor pero muy variado que va desde modelos que obtienen un 76% hasta un 87% de accuracy en la fase de entrenamiento; pero con valores muy bajos en la validación. A su vez se observa también que presenta un modelo con el kernel polinomial, cuyo desempeño en la fase de validación alcanza valores de 90% de accuracy y 95% de F1 score. Al analizar los resultados se puede ver que al considerar una validación cruzada con un valor *K= 5*, significa que el 20% de los objetos no se toman en cuenta para el entrenamiento, y algunos de ellos son representativos para las clases.

Analizando el tiempo del clasificador Random Forest obtiene un tiempo promedio de 640 ms en la fase de entrenamiento y 34 ms en la fase de prueba. En el entrenamiento, MVS logra en promedio 1.4 segundos y 1.53 segundos para cada kernel respectivamente, y con respecto a la fase de validación se presenta una disminución del tiempo promedio, sobre todo con el kernel polinomial que logra tiempos de 240 ms.

Tabla 4.7 Resultados de la clasificación con validación cruzada con K=5.

Imagen RGB					
K=5					
Random Forest					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
0.59781742	1	1	0.03563786	0.4150772	0
0.65386486	1	1	0.03423023	0.74386921	0
0.68113852	1	1	0.03302908	0.78110808	0.85748078
0.6416862	**1**	**1**	0.03359675	**0.835604**	**0.91044038**
0.62799001	1	1	0.03692698	0.82833787	0.90611028
SVM-POLY					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.06698418	0.8726158	0.91433807	0.19610167	0.36058129	0
1.43137574	0.78088102	0.86235915	0.23775291	0.57311535	0
1.31968808	0.81289737	0.83605253	0.21567988	0.72025431	0.83315276
1.62441778	**0.76203451**	**0.76700756**	0.29485774	**0.90917348**	**0.95242626**
1.59948921	0.76861944	0.77167824	0.30021024	0.87920073	0.93571774
SVM-RBF					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.17911887	0.87511353	0.91574755	0.73769045	0.43505904	0
1.59256411	0.78769301	0.86602665	0.94527125	0.47683924	0
1.41608644	0.81948229	0.83890578	0.86883783	0.72116258	0.832515
1.76806378	0.76771117	0.76509759	1.04821444	0.88737511	0.94032724
1.7291491	0.77679382	0.77479954	1.02432251	0.84287012	0.91473632

Finalmente, en la tabla 4.8 se muestran los resultados obtenidos con la validación cruzada de *10 divisiones*. Los resultados confirman que el clasificador Random Forest presenta un mejor valor en la etapa de entrenamiento con un 100% de

accuracy y F1 score. Pero nuevamente los rangos obtenidos en la fase de validación son bajos, incluso el primer modelo tiene un 37% de accuracy en esta fase; sin embargo, el último modelo alcanza un 89% de accuracy y 94% de F1 score. Esto puede deberse a que hay pocas instancias con algunos tonos de cubrebocas y por la aleatoriedad propia de la validación cruzada, no son considerados en el entrenamiento.

Por su parte, MSV, con ambos kernels, presenta en la fase de entrenamiento un desempeño variado, con modelos que varían del 77% al 83% de accuracy; no obstante, el rendimiento en la etapa de validación en algunos casos es bajo. Nuevamente, también se encuentran modelos cuyo desempeño en la fase de validación alcanzan valores de 90% de accuracy y 96% de F1 score, con el kernel polinomial.

Tabla 4.8 Resultados de la clasificación con validación cruzada con K=10.

Imagen RGB					
K10					
Random Forest					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
0.70416665	0.99979814	0.99983555	0.02709651	0.37205082	0
0.70761085	1	1	0.0265944	0.55716878	0
0.77360487	1	1	0.02439475	0.92377495	0
0.70632148	1	1	0.02618766	0.676951	0
0.73566771	1	1	0.02353024	0.66061706	0.74206897
0.72052193	1	1	0.02429008	0.86727273	0.92891918
0.72244716	1	1	0.02427125	0.88909091	0.9412897
0.72089529	1	1	0.02428222	0.88909091	0.9412897
0.70575762	1	1	0.02330256	0.88363636	0.93822394
0.70973802	**1**	**1**	0.02468491	**0.89818182**	**0.94636015**
SVM-POLY					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.63560009	0.82842148	0.87440898	0.18426895	0.29219601	0
1.68928385	0.82741219	0.87413514	0.14560819	0.46823956	0
1.95117807	0.77654421	0.84371029	0.15558314	0.90381125	0
1.77243567	0.81207105	0.86509202	0.16399693	0.61705989	0
1.62881041	0.82579734	0.85987985	0.12051511	0.54264973	0.68811881
1.91079068	0.77981837	0.81328085	0.15619731	0.93454545	0.96616541
1.97918534	0.7765893	0.81041274	0.16097617	**0.95272727**	**0.97579143**
1.97116876	0.7765893	0.81002231	0.16334009	0.96909091	0.98430286
1.90990973	0.77921292	0.81228552	0.22944117	0.93454545	0.96616541
2.15909433	0.77679112	0.80990031	0.16181469	0.96181818	0.98053753
SVM-RBF					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.78244209	**0.83366976**	**0.87788975**	0.48530126	0.44283122	0
1.8110795	0.83387162	0.87841631	0.49322581	0.47186933	0
2.10935807	0.7878482	0.85017819	0.61774826	0.8784029	0
1.90715337	0.82196205	0.87097718	0.50955677	0.61524501	0
1.92978024	0.83023819	0.86170038	0.47216296	0.52450091	0.67002519
2.17694688	0.79576186	0.82295311	0.55072141	0.92727273	0.96226415
2.10795546	0.7925328	0.81958582	0.5718317	**0.94545455**	**0.97196262**
2.11336565	0.79233098	0.81886992	0.56336236	0.95272727	0.97579143
2.08441997	0.79535822	0.8216045	0.56052446	0.92181818	0.95931883
2.20340419	0.79273461	0.81922197	0.55766582	0.93636364	0.96713615

Al igual que en los casos anteriores, se guardaron los tiempos de entrenamiento y test para cada modelo. En la tabla 4.8 se observa que Random Forest logra obtener un tiempo promedio de 720 ms en el entrenamiento mientras que MVS obtiene 1.8 a 2 segundos. En la fase de pruebas Random Forest obtuvo un tiempo promedio de 24 ms mientras que el menor tiempo promedio de MVS es de 164 ms con el kernel polinomial.

Caso B: Resultados de la Clasificación Con Preprocesamiento

En la tabla 4.9 se pueden observar los resultados de la clasificación 80/20 obtenidos con las métricas accuracy y F1 score. Se obtienen muy buenos resultados en el entrenamiento con el clasificador Random Forest, logrando un 100% en ambas métricas; sin embargo, su desempeño baja en la etapa de validación alcanzando solo el 83% de accuracy y 86% de F1 score.

El clasificador MVS obtiene una mejor respuesta con el kernel Polinomial sin lograr superar a Random Forest al obtener un accuracy de 78% de entrenamiento y un 77% de pruebas. Los resultados que se obtuvieron con el kernel rbf son ligeramente superiores en la etapa de validación.

El tiempo de entrenamiento y validación también fueron considerados en este caso. Se observa que Random Forest se entrena en 680 milisegundo a diferencia de MVS que superan los 2.2 segundos en entrenamiento (con ambos kernels). Se hace notar que MVS con el kernel polinomial muestra una disminución de su tiempo en la fase de prueba, sin lograr superar a Random Forest que obtiene un tiempo de 189 ms en el test.

Tabla 4.9 Resultado de la clasificación considerando el porcentaje de los datos a 80/20.

Imagen YCrCb					
80/20					
Random Forest					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
0.68029404	**1**	**1**	0.18987846	**0.83865087**	**0.86052009**
SVM-POLY					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
2.21031284	**0.78122151**	**0.82696467**	0.59145474	0.77301732	0.82047585
SVM-RBF					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
2.56973863	0.77666363	0.82386772	1.34463024	**0.7739289**	**0.82028986**

En la tabla 4.10 se muestran los resultados obtenidos con la validación cruzada de 5 divisiones. Se observa, nuevamente que el clasificador Random Forest obtiene el mejor desempeño en el entrenamiento con un 100% de accuracy y F1 score. Para la fase de prueba se observa que baja su desempeño, alcanzando un 84% de accuracy y 91% de F1 score.

Por parte del clasificador MVS tiene un desempeño menor, con ambos kernels, incluso obtienen un 73% de accuracy en la fase de entrenamiento con un desempeño muy bajo en la validación. A su vez se encuentran modelos cuyo desempeño en la validación alcanzan valores de accuracy de 84.86% hasta un 91.81% de F1 score con el kernel Polinomial. Analizando los resultados se observa que al considerar una validación cruzada de *5 divisiones* se forman subconjuntos de datos que no consideran objetos representativos para cada clase y que afectan a los modelos obtenidos.

Analizando el tiempo del clasificador Random Forest, obtiene un tiempo promedio de 654 ms en la fase de entrenamiento y 34 ms en la fase de prueba. En el entrenamiento, MVS logra en promedio 1.470 segundos y 1.64 segundos para cada kernel; y en la validación logra una disminución del tiempo promedio, sobre todo con el kernel polinomial con 260 ms.

Tabla 4.10 Resultados de la clasificación con validación cruzada con K=5.

Imagen YCrCb					
K5					
Random Forest					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
0.61269808	0.99977211	0.99983555	0.03328443	0.4402917	0
0.65726352	1	1	0.03405595	0.70009116	0
0.70184684	1	1	0.03528619	0.77848678	0.85334943
0.66978836	**1**	**1**	0.03675199	**0.84503191**	**0.91600791**
0.63302445	1	1	0.03444505	0.79033728	0.88289206
SVM-POLY					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.14954925	0.8452598	0.89750943	0.18783569	0.32087511	0
1.45205474	0.74247949	0.84318623	0.2308259	0.08021878	0
1.46561384	0.80355515	0.82585859	0.26372409	0.73290793	0.83945205
1.65950203	**0.73974476**	**0.73454207**	0.3148694	**0.84867821**	**0.91814596**
1.62604117	0.74658159	0.73798303	0.30826831	0.7538742	0.85966736
SVM-RBF					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.29471397	0.8411577	0.89523523	0.82463408	0.35642662	0
1.53464055	0.74316317	0.84266369	0.91250491	0.04193254	0
1.61872911	0.80355515	0.82429678	0.97302818	0.73290793	0.83927592
1.90329313	0.74407475	0.73804525	1.11898971	0.84685506	0.91707799
1.87128854	0.75638104	0.74817432	1.07680392	0.76937101	0.86965482

Finalmente, en la tabla 4.11 se muestran los resultados obtenidos con la validación cruzada con *10 divisiones*. Los resultados confirman que con el clasificador Random Forest presenta un mejor valor en la etapa de entrenamiento con un 100% de accuracy y F1 score. De manera similar a la validación cruzada de *5 divisiones*, en la etapa de validación se obtienen desempeños bajos en los primeros modelos; sin embargo, se alcanza un 90% de accuracy y 95% de F1 *score* con el modelo mostrado en el renglón 8. Esto podría deberse a la aleatoriedad que se da en la formación de los subconjuntos de imágenes y que algunos tonos de cubrebocas están presentes, pero con pocas instancias para el entrenamiento.

Con respecto al clasificador MVS, como en los casos anteriores presenta un desempeño variado con ambos kernels, con valores que varían del 76% al 80% de accuracy; y el rendimiento en la validación en algunos casos es bajo. Nuevamente, también se encuentran modelos cuyo desempeño en la fase de validación alcanzan valores de 88% de accuracy y 94% de F1 score, con el kernel polinomial.

Con respecto al tiempo de respuesta, en la tabla 4.11 se observa que Random Forest logra un tiempo promedio de 730 Ms en el entrenamiento mientras que MVS obtiene de 2 a 2.14 segundos. En la fase de pruebas Random Forest obtuvo un tiempo promedio de 25 ms mientras que el menor tiempo promedio de MVS es de 166 ms con el kernel polinomial.

Tabla 4.11 Resultados de la clasificación con validación cruzada con K=10.

Imagen YCrCb					
K10					
Random Forest					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
0.74523687	1	1	0.02409959	0.52276867	0
0.73809123	1	1	0.02504659	0.55191257	0
0.76383066	1	1	0.02542591	0.9143898	0
0.71447277	1	1	0.02864528	0.70856102	0
0.7396915	1	1	0.02419543	0.72131148	0.76569678
0.71194148	1	1	0.0251317	0.85036496	0.91913215
0.7605207	1	1	0.02416158	0.8850365	0.93901258
0.73368883	**1**	**1**	0.02515459	**0.90510949**	**0.95019157**
0.72611713	1	1	0.02402472	0.85036496	0.91913215
0.73762465	1	1	0.0297482	0.8649635	0.92759295
SVM-POLY					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.79542089	**0.80733387**	**0.86000294**	0.14897513	0.37522769	0
1.76914787	0.80773906	0.86119643	0.15090775	0.43897996	0
2.02407885	0.75850891	0.83258427	0.1730485	0.75045537	0
1.89651108	0.79517828	0.85311637	0.15379667	0.56466302	0
1.78360248	0.81300648	0.84896089	0.14851213	0.5701275	0.70426065
2.05854774	0.76990075	0.80077166	0.17633414	0.88868613	0.9410628
2.11087871	0.76524205	0.79598662	0.17785931	0.92335766	0.9601518
2.42048812	0.76362163	0.79508341	0.18103313	**0.94343066**	**0.97089202**
2.13272953	0.76726757	0.79760437	0.1775341	0.88868613	0.9410628
2.038203	0.7650395	0.79519774	0.1771946	0.89963504	0.94716619
SVM-RBF					
Time TRAIN	Accuracy	Train F1	Time TEST	Accuracy	Test F1
1.96958041	0.80307942	0.85701677	0.58322239	0.49180328	0
1.93608141	0.80490276	0.8590663	0.56933475	0.43715847	0
2.26592135	0.75182334	0.82835925	0.56845903	0.67577413	0
2.02391958	0.7886953	0.84908117	0.53498912	0.54098361	0
1.98251438	0.8123987	0.8475971	0.51133609	0.57741348	0.71
2.22210908	0.76665991	0.79761068	0.59379601	0.89963504	0.94716619
2.26449919	0.76260887	0.79351656	0.59321737	0.91423358	0.95519542
2.295156	0.76200122	0.79367867	0.60076499	**0.9270073**	**0.96212121**
2.24016881	0.76483695	0.7952742	0.59986472	0.89051095	0.94208494
2.25478268	0.76362163	0.79421619	0.58943033	0.89051095	0.94208494

4.2.4 Caso 4: Medición del Tiempo de Respuesta

Para calcular el tiempo promedio de respuesta del sistema, se realizó un video que presenta a tres personas con cambios en el color de cubrebocas, en la calidad de

la imagen, de iluminación y en diversos días y, con ello con una variedad en las condiciones del entorno, así como presencia de cabello en la región de la frente.

El video está en formato .mp4, tiene una duración de 2 min 24 segundos que equivale a 1,756 fotogramas; en la figura 4.7 se pueden observar algunas capturas de los entornos presentes en el video.

Figura 4.7 Imágenes muestra del video de prueba creado.

El video se dio como entrada al sistema final que se integra de las técnicas presentadas en la figura 1 sección 3. El video obtuvo un tiempo promedio de 0.21168 ms para cada fotograma del video analizado; sin embargo, es importante mencionar que este tiempo puede variar dependiendo de las características de la computadora donde se quiera correr el sistema.

4.3 Análisis de Resultados

En el caso de estudio 1 se observó que la mascarilla provoca que los rostros tengan una oclusión mayor al cubrir aproximadamente la mitad de la cara. De acuerdo a los resultados obtenidos se encontró que MediaPipe es la herramienta más robusta en la detección de rostros con oclusiones, siendo capaz de localizar el rostro en condiciones extremas de poca iluminación, baja escala y con una mayor rotación.

Con respecto al caso de estudio 2 se concluye que el descriptor LBP muestra mejores resultados en la mayoría de los casos de entrenamiento con SVM. LBP También tuvo una buena respuesta con Random Forest en combinación con la normalización de 0-255, obteniendo un porcentaje superior al 90% de accuracy y una precisión por arriba del 96%.

Sin embargo, el descriptor BSIF ha demostrado que puede lograr una buena respuesta de clasificación también, obteniendo un accuracy por arriba del 90% con el clasificador SVM.

Con respecto al caso de estudio 3, note que Random Forest es constante en los resultados con los datos de entrenamiento y prueba, obteniendo un valor de accuracy de 100% y un 100% de F1 score y hasta un 90% de accuracy y 95% de F1 *score* en la fase de validación. Mientras que el rendimiento del clasificador SVM (con ambos kernels) baja considerablemente. Obtiene un mejor accuracy y F1 score por arriba del 80% en entrenamiento y un mejor accuracy y F1 score arriba del 90% en test sin embargo no hay equilibrio entre el entrenamiento y el test.

Revisando en detalle los resultados con y sin procesamiento, el clasificador Random Forest muestra un desempeño robusto cuya variación es menor. De hecho, aplicando el preprocesamiento en la imagen se logra hasta un 90% de accuracy y 95% de F1 score de test con la validación cruzada con 10 divisiones. Por lo cual, se concluye que es el modelo con mejor rendimiento y se considera para el sistema final.

Finalmente, para el caso 4 se concluye que se obtuvo un tiempo promedio de 0.21168 ms para cada fotograma del video analizado, en una computadora de características normales (ver anexo C).

4.4 Comparativa con el Estado del Arte

Como ya se mencionó, debido a la pandemia identificar que una persona utiliza la presencia el cubrebocas, a partir del análisis automático de una imagen, se volvió un área de investigación reciente y de mucho interés. Por ello, como última parte del análisis de los resultados se muestra en la tabla 4.12 una comparativa de algunos artículos reportados en el estado del arte y el trabajo de tesis detallado en este reporte.

Tabla 4.12 Comparativa con el estado del arte.

Referencia	Enfoque	Herramientas	Conjuntos	Resultados
Deep Learning				
(Bu et al., 2017)	Deep learning.	Clasificadores CNN.	MASKED FACE, WIDER FACE.	Se evaluó con PASCAL VOC obteniendo 86.6% de precisión y 87.7% en la recuperación.
(Wang et al., 2020)	Deep learning.	Modelo de granularidad múltiple basado en ojos y el rostro, Dlib.	MFDD, RMFRD, SMFRD.	Alcanzo una precisión del 95%.
(Hariri, 2020)	Deep learning.	VGG-16, Librería Dlib-ml.	RMFRD.	La mejor tasa de reconocimiento fue del 91.3%.

Tabla 4.12 Comparativa con el estado del arte.

Referencia	Enfoque	Herramientas	Conjuntos	Resultados
Deep Learning				
(sixphere et al., 2020)	Deep learning.	OpenCV, Python.	Conjunto de 20,000 imágenes.	Implementando en PC se logró una detección de 70 ms mientras que en una Raspberry fue de 270ms.
(Mucha et al., 2021)	Deep learning.	Python, VGG16, Haar cascade frontal face, NumPy. Red.h5.	2,400 imágenes.	Reportan un accuracy de 71% en la identificación de personas en un tiempo aproximado de 10 segundos por detección.
(Cayetano, 2021)	Deep learning.	Python, TensorFlow, Keras, ReLu, SoftMax.	700 imágenes propias.	96% de precisión en el reconocimiento del cubrebocas.
(Cabani et al., 2021)	Deep learning.	Clasificadores en cascada con características de Haar	MaskedFace-Net.	MaskedFace-Net integra CMFD en 49% y IMFD en 51%.
Visión Artificial Clásico				
(Cutipa et al., 2019)	Visión artificial y aprendizaje de máquinas.	SVM, PCA, Clasificador cascada.	FERET, ORL y Propio.	Se menciona que su sistema obtuvo un 98% de precisión.
(Mundial et al., 2020)	Visión artificial y aprendizaje de máquinas.	Aprendizaje automático, clasificador SVM.	VGGFACE2.	Con SVM obtuvieron hasta 97% y hasta 98% de precisión al incluir poses laterales del rostro.
Trabajo Propio	Visión artificial y aprendizaje de máquinas.	YCrCb, MediaPipe, LBP, Random Forest.	MaskedFace-Net, RMFRD, Flickr-Faces-HQ (FFHQ), Propio	Se obtiene un 90% de accuracy y un 95% de F1 score, con un tiempo promedio de 211 ms en la respuesta del sistema.

Como se puede ver, el presente proyecto logra resultados semejantes o ligeramente abajo a los reportados. Sin embargo, el presente proyecto es más robusto en la localización del rostro al utilizar la herramienta de MediaPipe en contra Dlib que utiliza la mayoría de los artículos analizados y con ello el desempeño del sistema ante la variabilidad del medio ambiente es más robusto a los reportados en el estado del arte. Además, a diferencia del enfoque de Deep Learning, en el presente proyecto se conoce manera clara a los descriptores que participan en la representación del objeto.

Capítulo 5

Conclusiones

5.1 Conclusiones Generales

En el presente proyecto de tesis se desarrolló un sistema que, mediante el análisis automático de las imágenes introducidas, puede identificar si la persona porta un cubrebocas o no, y si además la mascarilla cubre la nariz y la boca. El sistema es robusto a cambios en entornos no controlados; es decir, con diferente escala, rotación, intensidad luminosa, oclusión causada por prendas o complementos del vestuario como pueden ser sombreros, gorras, cascos, lentes graduados, cabello, incluso maquillaje, diversos tonos de piel y, sin olvidar la oclusión del rostro ocasionada por el uso del cubrebocas y diversos colores del mismo (azul, blanco, negro, rosa y morado).

También se puede concluir que el aplicar a la imagen de entrada el preprocesamiento de ecualizar la banda *Y*, en el modelo de color YCbCr, puede mejorar ligeramente la predicción de las tres regiones de interés: frente, nariz y boca. Sin embargo, trabajar la imagen original en formato RGB (sin dicho preprocesamiento) también proporciona buenos resultados en un tiempo promedio de predicción de 0.2116 ms.

Con respecto a las herramientas de localización de rostros, se puede concluir que Dlib es una de las herramientas más utilizada en la literatura al tener buenos resultados en la detección de rostros sin presencia de oclusión, pero tiene problemas con imágenes escasas de luminosidad y cuando no puede localizar ambos ojos. En el caso de DNN genera falsos positivos cuando el rostro cubre casi por completo la imagen; MTCNN no presenta la detección de falsos positivos; sin embargo, es el menos preciso en la detección del rostro cuando existe algún tipo de oclusión como el uso del cubrebocas. Por su parte MediaPipe es una herramienta relativamente nueva que ha alcanzado gran importancia gracias a sus múltiples aplicaciones en diversas plataformas y tiene la suficientemente robustez para realizar la tarea de detectar y localizar el rostro ocluidos y en entornos no controlados obteniendo una precisión del 97.5%, una exactitud del 86.7% y un 92.9 % de F1 score en este trabajo.

También se realizó una comparativa de los descriptores de textura LBP y BSIF. De acuerdo a la experimentación realizada, se seleccionó el descriptor LBP para el

sistema final al mostrar una buena respuesta, sin bajar el rendimiento entre las etapas de entrenamiento y validación. Obteniendo un porcentaje superior al 90% de accuracy y una precisión por arriba del 96% con Random Forest con la normalización de *0-255* (ver sección 4.2).

Con respecto a la clasificación de la sección 4.3, se obtuvieron los mejores resultados con el clasificador Random Forest superando a SVM con los dos kernels considerados en el trabajo. En los tres experimentos realizados, Random Forest obtuvo un 100% de accuracy y F1 Score en entrenamiento y desde un 83% hasta un 90.51% de accuracy y un 86% hasta un 95.01% de F1 score en la etapa de validación. Por ello, el sistema final considera el antepenúltimo modelo de la validación cruzada con *k=10* de la imagen con pre procesamiento para identificar el uso de cubrebocas (sección 4.2.3, tabla 4.11).

5.2 Objetivos Logrados

En la tabla 5.1 se listan los objetivos del proyecto y una breve descripción de cómo se trabajó para cumplirlos.

Tabla 5.1 Objetivos logrados.

Objetivo General	
Objetivos	**Solución del objetivo**
"Diseñar y desarrollar un sistema que identifique si una persona porta una mascarilla en el rostro y se encuentra puesta de forma correcta, mediante técnicas de visión artificial y aprendizaje automático."	**Logrado:** Se diseñó y desarrolló en Python un sistema de visión artificial que, mediante el análisis de las imágenes, identifica si una persona porta una mascarilla en el rostro y se encuentra puesta de forma correcta. El sistema alcanza hasta un 90% de accuracy y 95% de F1 con el clasificador Random Forest.
Objetivos Específicos	
Objetivos	**Solución del objetivo**
"Analizar el estado del arte en las áreas de reconocimiento de personas con rostro ocluido y detección de mascarillas en el rostro; además de revisar las técnicas de visión artificial y aprendizaje automático con mejor desempeño."	**Logrado:** Se analizaron 70 artículos y se reportan 35 artículos en el estado del arte referentes a este trabajo. En las categorías de detección de mascarilla, reconocimiento facial, detección del rostro con oclusiones, reconocimiento facial con oclusiones.
"Proponer y desarrollar un sistema que identifique cuando una persona porta el cubrebocas cubriendo nariz y boca proporcionando la respuesta en el menor tiempo posible."	**Logrado:** El sistema logra detectar el uso correcto del cubrebocas en un ambiente real, en un tiempo promedio de 0.2116 ms en una laptop Asus X515JA con Windows 10 versión 21H1 64 bits, con procesador Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz, disco de estado sólido/EMMC1 256GB, memoria RAM 12GB.

Tabla 5.1 Objetivos logrados.

Objetivos Específicos	
Objetivos	**Solución del objetivo**
"Evaluar el sistema desarrollado con las métricas tradicionales de la clasificación supervisada."	**Logrado:** El sistema fue evaluado con 4 métricas clásicas en el área del aprendizaje de máquinas: Exactitud (Accuracy), Sensibilidad (Recall), Precisión, F1 score. En las tablas de resultados se reportan solamente dos; sin embargo, para obtener la métrica F1 score son necesarias la Sensibilidad (Recall) y Precisión.

5.3 Aportaciones

1. La principal aportación de este trabajo es el desarrollo de un sistema de visión artificial robusto para el reconocimiento de la mascarilla facial bien puesta, esto significa que debe de cubrir la frente, nariz y boca. El sistema presenta un buen desempeño aun cuando la imagen presente rostros con características como las descritas en el punto 4 de esta sección.

2. El sistema entrega una respuesta en un tiempo promedio de 0.21168 ms.

3. Este sistema tiene la característica de poder trabajar con tres modalidades:
   - Video (.avi,.mp4)
   - Cámara web (predefinida del equipo donde se ejecuta el sistema)
   - Imágenes (.jpeg,.jpg,.png)

   Al finalizar el análisis en cualquiera de las tres modalidades se guarda el video o la imagen resultante.

4. El sistema puede localizar el rostro ante la presencia de:
   - Cubrebocas
   - Lentes de lectura (graduados) y/o protección de luz azul
   - Cabello en la región de la frente
   - Diversos tonos de piel
   - Diversa Escala del rostro
   - Ligera rotación del rostro analizado (requiere visibilidad de los ojos)
   - Cambios de entorno
   - Cambios en la iluminación (se requiere visibilidad del rostro en la imagen analizada)
   - Identifica diferentes tonos de cubrebocas tales como azul, blanco, negro, rosa y morado.

5. Detecta varios rostros presentes en la imagen, pero analiza la cara que detecta primero; generalmente, la que cubre una mayor área de la imagen.

6. Se creó un conjunto de 583 imágenes de por lo menos 3 personas, adquiridas en diferentes días y entornos lo que provoca cambios:
   - De iluminación
   - De rotación del rostro
   - Variabilidad en el tipo y colores del cubrebocas en condiciones reales

5.4 Trabajos Futuros

Se considera importante mejorar la imagen de entrada con filtros de luminosidad para tener una mejor visibilidad del rostro presente en la imagen.

Debido a la variabilidad de los cubrebocas, se considera aumentar la detección del mismo considerando diversos tonos e incluso presencia de figuras o bordados; además se pretende incrementar el número de imágenes del conjunto propio con mayor variedad del tono del cubrebocas en entornos reales no controlados.

Durante el desarrollo de la herramienta se revisaron y encontraron diversos descriptores de textura, pero se seleccionaron dos de los mejores de acuerdo a los reportes de la literatura, pero se considera que se pueden evaluar otros descriptores de textura a los implementados en este trabajo.

5.5 Actividades Académicas Adicionales

Se presentó (de manera virtual) el artículo “Comparativa de Herramientas para Localizar Rostros Ocluidos” en el marco de la 8ª jornada de ciencia y tecnología aplicada, que se celebró del 22 al 25 de mayo de 2022 en el TecNM/CENIDET, ver figura 5.1.

EDUCACIÓN

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL CENTRO NACIONAL DE INVESTIGACIÓN
Y DESARROLLO TECNOLÓGICO

OTORGA EL PRESENTE

RECONOCIMIENTO

A

JOSE OMAR DE JESUS TRUJILLO QUINTERO, ANDREA MAGADÁN SALAZAR, JONATHAN VILLANUEVA TAVIRA AND RAÚL PINTO ELÍAS

POR LA PRESENTACIÓN DEL ARTICULO:
COMPARATIVA DE HERRAMIENTAS PARA LOCALIZAR ROSTROS OCLUIDOS
EN EL MARCO DE LA 8ª JORNADA DE CIENCIA Y TECNOLOGÍA APLICADA, CELEBRADO
DEL 25 AL 27 DE MAYO DE 2022, EN EL TECNM/CENIDET.

CUERNAVACA, MORELOS., 27 DE MAYO DE 2022

DRA. YESICA IMELDA SAAVEDRA BENÍTEZ
DIRECTORA DEL CENTRO NACIONAL DE INVESTIGACIÓN
Y DESARROLLO TECNOLÓGICO

Sello Digital:

cenidet

Figura 5.1 Reconocimiento 8ª jornada de ciencia y tecnología aplicada.

Presentación del poster “Sistema de Visión Artificial para el Reconocimiento de la Mascarilla Facial Bien Puesta” en la escuela de inteligencia computacional y robótica 2022, realizada en la Universidad Tecnológica Emiliano Zapata (UTEZ), los días 16 y 20 de agosto, ver figura 5.2.

IEEE Computational Intelligence Society
IEEE Robotics & Automation Society
IEEE

El Capítulo de Inteligencia Computacional y Robótica del IEEE de la Sección Morelos

Certifica que:

Jose Omar de Jesus Trujillo Quintero

Participó en la Exposición de Pósters con el trabajo titulado:

"Sistema de Visión Artificial para el Reconocimiento de la Mascarilla Facial Bien Puesta"

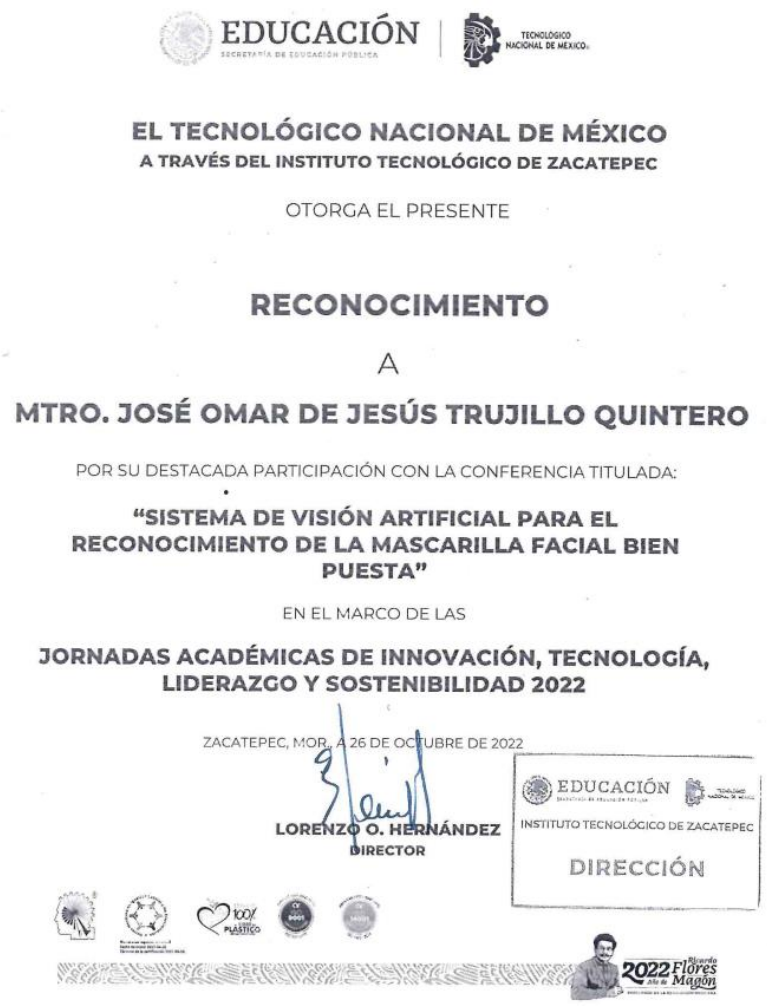
Para acceso universal al conocimiento y sus beneficios sociales en el marco de la Escuela de Inteligencia Computacional y Robótica realizada del 16 al 20 de Agosto de 2022.

Dr. Jonathan Villanueva Tavira
Presidente del Capítulo

Dra. Andrea Magadán Salazar
Presidenta del Programa Técnico

Figura 5.2 Reconocimiento escuela de inteligencia computacional y robótica 2022.

Se presentó (de manera virtual) la ponencia “Sistema de Visión Artificial para el Reconocimiento de la Mascarilla Facial Bien Puesta” en el marco de las Jornadas Académicas de Innovación, Tecnología, Liderazgo y Sostenibilidad 2022, que se celebró el 26 de octubre de 2022 en TecNM/Instituto Tecnológico de Zacatepec, ver figura 5.3.


EDUCACIÓN

EL TECNOLÓGICO NACIONAL DE MÉXICO
A TRAVÉS DEL INSTITUTO TECNOLÓGICO DE ZACATEPEC

OTORGA EL PRESENTE

RECONOCIMIENTO

A

MTRO. JOSÉ OMAR DE JESÚS TRUJILLO QUINTERO

POR SU DESTACADA PARTICIPACIÓN CON LA CONFERENCIA TITULADA:

“SISTEMA DE VISIÓN ARTIFICIAL PARA EL RECONOCIMIENTO DE LA MASCARILLA FACIAL BIEN PUESTA”

EN EL MARCO DE LAS

JORNADAS ACADÉMICAS DE INNOVACIÓN, TECNOLOGÍA, LIDERAZGO Y SOSTENIBILIDAD 2022

ZACATEPEC, MOR., A 26 DE OCTUBRE DE 2022

LORENZO O. HERNÁNDEZ
DIRECTOR

EDUCACIÓN
INSTITUTO TECNOLÓGICO DE ZACATEPEC
DIRECCIÓN

2022 Ricardo Flores Magón

Figura 5.3 Reconocimiento jornadas académicas de innovación, tecnología, liderazgo y sostenibilidad 2022.

Presentación del poster “Sistema de Visión Artificial para el Reconocimiento de la Mascarilla Facial Bien Puesta” en el marco de la 9ª jornada de ciencia y tecnología aplicada, que se celebró del 16 al 18 de noviembre de 2022 en el TecNM/CENIDET, ver figura 5.4.

Figura 5.4 Reconocimiento 9ª jornada de ciencia y tecnología aplicada.

Referencias

Ahonen, T., Hadid, A., & Pietikainen, M. (2006). Face description with local binary patterns: Application to face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *28*(12), 2037–2041.

Alexe, B., Deselaers, T., & Ferrari, V. (2010). What is an object? *2010 IEEE Computer Society Conference on Computer Vision and Pattern Recognition*, 73–80.

Alpaydin, E. (2014). *Introduction to Machine Learning, 3rd Editio. ed*. The MIT Press.

Anisimov, D., & Khanova, T. (2017). Towards lightweight convolutional neural networks for object detection. *2017 14th IEEE International Conference on Advanced Video and Signal Based Surveillance (AVSS)*, 1–8.

Arandjelovic, R., & Zisserman, A. (2013). All about VLAD. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 1578–1585.

Assi, A., Shanwar, B., & Zarka, N. (2016). *Detection of Abandoned Objects in Crowded Environments*. https://doi.org/10.13140/RG.2.1.3884.6963

Azeem, A., Sharif, M., Raza, M., & Murtaza, M. (2014). A survey: Face recognition techniques under partial occlusion. *International Arab Journal of Information Technology*, *11*(1), 1–10.

Barrios, J. (2020, June 7). *Algoritmo para la detección de mascarilla*. https://www.juanbarrios.com/algoritmo-para-la-deteccion-del-uso-de-la-mascarilla/

Barrios, J. I. (2019, June 26). *La matriz de confusión y sus métricas – Inteligencia Artificial –*. https://www.juanbarrios.com/la-matriz-de-confusion-y-sus-metricas/

Bazarevsky, V., Kartynnik, Y., Vakunov, A., Raveendran, K., & Grundmann, M. (2019). *BlazeFace: Sub-millisecond Neural Face Detection on Mobile GPUs*. 3–6. http://arxiv.org/abs/1907.05047

Benzaoui, A., Adjabi, I., & Boukrouche, A. (2017). Experiments and improvements of ear recognition based on local texture descriptors. *Optical Engineering*, *56*(4), 43109.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. *Proceedings of the Fifth Annual Workshop on Computational Learning Theory*, 144–152.

Bu, W., Xiao, J., Zhou, C., Yang, M., & Peng, C. (2017). A cascade framework for masked face detection. *2017 IEEE International Conference on Cybernetics and Intelligent Systems, CIS 2017 and IEEE Conference on Robotics,*

Automation and Mechatronics, RAM 2017 - Proceedings, *2018-Janua*, 458–462. https://doi.org/10.1109/ICCIS.2017.8274819

Cabani, A., Hammoudi, K., Benhabiles, H., & Melkemi, M. (2021). MaskedFace-Net – A dataset of correctly/incorrectly masked face images in the context of COVID-19. *Smart Health*, *19*, 100144. https://doi.org/10.1016/j.smhl.2020.100144

Cao, Q., Shen, L., Xie, W., Parkhi, O. M., & Zisserman, A. (2018). VGGFace2: A dataset for recognising faces across pose and age. *Proceedings - 13th IEEE International Conference on Automatic Face and Gesture Recognition, FG 2018*, 67–74. https://doi.org/10.1109/FG.2018.00020

Cayetano, I. R. P. (2021). Detección automática de rostros con cubreboca o sin cubreboca para restringir el acceso a institución educativa. *Revista Aristas*, *8*(16), 154–160.

Chang, C.-C. (2011). " LIBSVM: a library for support vector machines," ACM Transactions on Intelligent Systems and Technology, 2: 27: 1--27: 27, 2011. *Http://Www. Csie. Ntu. Edu. Tw/\~{} Cjlin/Libsvm*, *2*.

Chaves, D., Saikia, S., Fernández-Robles, L., Alegre, E., & Trujillo, M. (2018). Una revisión sistemática de métodos para localizar automáticamente objetos en imágenes. *Revista Iberoamericana de Automática e Informática Industrial*, *15*(3), 231–242.

Comon, P. (1994). Independent component analysis, a new concept? *Signal Processing*, *36*(3), 287–314.

Cornejo, J. Y. R., & Pedrini, H. (2016). Recognition of occluded facial expressions based on CENTRIST features. *2016 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, 1298–1302.

Cutipa, R. A., Sánchez, G. F. C., Quispe, V. I., & Figueroa, E. N. T. (2019). Facial detection and recognition system using hybrid techniques in images and video sequences. *Revista de Investigación C\&T Riqchary*, *1*(1), 41–46.

Dalal, N., & Triggs, B. (2005). Histograms of oriented gradients for human detection. *2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR'05)*, *1*, 886–893.

Dlib. (2022, January 24). *dlib C++ Library*. http://dlib.net/

Everingham, M., Van Gool, L., Williams, C. K. I., Winn, J., & Zisserman, A. (2010). The pascal visual object classes (voc) challenge. *International Journal of Computer Vision*, *88*(2), 303–338.

Felzenszwalb, P. F., Girshick, R. B., McAllester, D., & Ramanan, D. (2010). Object detection with discriminatively trained part-based models. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, *32*(9), 1627–1645.

Garcia Rios, E., Escamilla Hernandez, E., Nakano Miyatake, M., & Perez Meana, H. (2017). Face Recognition with Occlusion using a wireframe model and Support Vector Machine. *IEEE Latin America Transactions*, *15*(10), 1960–1966. https://doi.org/10.1109/TLA.2017.8071241

Ge, S., Li, J., Ye, Q., & Luo, Z. (2017). Detecting masked faces in the wild with LLE-CNNs. *Proceedings - 30th IEEE Conference on Computer Vision and Pattern Recognition, CVPR 2017*, *2017-Janua*, 426–434. https://doi.org/10.1109/CVPR.2017.53

Girshick, R. B. (2015). *Fast R-CNN. CoRR, abs/1504.08083*.

Girshick, R., Donahue, J., Darrell, T., & Malik, J. (2013). Rich feature hierarchies for accurate object detection and semantic segmentation. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 580–587.

Golwalkar, R., & Mehendale, N. (2020). Masked Face Recognition Using Deep Metric Learning and FaceMaskNet-21. *Available at SSRN 3731223*.

Gonzalez-Sosa, E., Vera-Rodriguez, R., Fierrez, J., & Ortega-Garcia, J. (2016). Validacion Experimental de la Influencia de Oclusiones en Reconocimiento Facial. *Proc. of The XXXI Simposium Nacional de La Unión Científica Internacional de Radio (URSI)*.

González, R. C., & Woods, R. E. (1996). *Tratamiento digital de imágenes* (Vol. 3). Addison-Wesley New York.

Gonzalez, R. C., Woods, R. E., & Eddins, S. L. (2009). *Digital Image Processing Using MATLAB* (2nd ed.). GatesMark Publishing.

Goodfellow, I., Warde-Farley, D., Mirza, M., Courville, A., & Bengio, Y. (2013). Maxout networks. *International Conference on Machine Learning*, 1319–1327.

Hariri, W. (2020). *Efficient Masked Face Recognition Method during the COVID-19 Pandemic*.

He, K, Gkioxari, G., Dollár, P., & Girshick, R. B. (2017). Mask R-CNN. CoRR abs/1703.06870. *URL: Http://Arxiv. Org/Abs/1703.06870*.

He, Kaiming, Zhang, X., Ren, S., & Sun, J. (2016). Deep residual learning for image recognition. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 770–778.

Herta lanza un reconocimiento facial que permite identificar hasta con mascarilla. (2020, March 10). https://www.prnewswire.com/news-releases/herta-lanza-un-reconocimiento-facial-que-permite-identificar-hasta-con-mascarilla-872615280.html

Hu, J., Shen, L., & Sun, G. (2018). Squeeze-and-excitation networks. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 7132–7141.

Huang, B., Wang, Z., Wang, G., Jiang, K., He, Z., Zou, H., & Zou, Q. (2021). Masked Face Recognition Datasets and Validation. *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) Workshops*, 1487–1491.

Huang, G. B., & Learned-Miller, E. (2014). Labeled faces in the wild: Updates and new reporting procedures. *Dept. Comput. Sci., Univ. Massachusetts Amherst, Amherst, MA, USA, Tech. Rep*, *14*(003).

Kannala, J., & Rahtu, E. (2012). BSIF: Binarized statistical image features.

Proceedings - International Conference on Pattern Recognition, *Icpr*, 1363–1366.

Kavitha, K. R., Vijayalakshmi, S., Annakkili, A., Aravindhan, T., & Jayasurya, K. (2021). Face Mask Detector Using Convolutional Neural Network. *Annals of the Romanian Society for Cell Biology*, 1979–1985.

Kawamoto, S., Shimodaira, H., Nitta, T., Nishimoto, T., Nakamura, S., Itou, K., Morishima, S., Yotsukura, T., Kai, A., Lee, A., & others. (2004). Galatea: Open-source software for developing anthropomorphic spoken dialog agents. In *Life-like characters* (pp. 187–211). Springer.

King, D. E. (2009). Dlib-ml: A machine learning toolkit. *The Journal of Machine Learning Research*, *10*, 1755–1758.

Lampert, C. H., Blaschko, M. B., & Hofmann, T. (2008). Beyond sliding windows: Object localization by efficient subwindow search. *2008 IEEE Conference on Computer Vision and Pattern Recognition*, 1–8.

Learned-Miller, E., Huang, G. B., RoyChowdhury, A., Li, H., & Hua, G. (2016). Labeled faces in the wild: A survey. In *Advances in face detection and facial image analysis* (pp. 189–248). Springer.

Li, S., Deng, W., & Du, J. (2017). Reliable crowdsourcing and deep locality-preserving learning for expression recognition in the wild. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2852–2861.

Li, Y., Zeng, J., Shan, S., & Chen, X. (2018). Patch-Gated CNN for Occlusion-aware Facial Expression Recognition. *Proceedings - International Conference on Pattern Recognition*, *2018-Augus*, 2209–2214. https://doi.org/10.1109/ICPR.2018.8545853

Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C.-Y., & Berg, A. C. (2016). Ssd: Single shot multibox detector. *European Conference on Computer Vision*, 21–37.

Loey, M., Manogaran, G., Taha, M. H. N., & Khalifa, N. E. M. (2021). A hybrid deep transfer learning model with machine learning methods for face mask detection in the era of the COVID-19 pandemic. *Measurement: Journal of the International Measurement Confederation*, *167*(May 2020), 108288. https://doi.org/10.1016/j.measurement.2020.108288

López Ortiz, E., & López Ortiz, G. (2020). Uso generalizado de cubrebocas frente a la pandemia ocasionada por el SARS-CoV-2. *Atención Familiar*, *28*(1), 1. https://doi.org/10.22201/fm.14058871p.2021.1.77652

Lugaresi, C., Tang, J., Nash, H., Mcclanahan, C., Uboweja, E., Hays, M., Zhang, F., Chang, C., Yong, M. G., Lee, J., Chang, W., Hua, W., Georg, M., & Grundmann, M. (2019). *MediaPipe : A Framework for Perceiving and Augmenting Reality*.

Magadán, A. S. (1999). Reconocimiento de rostros invariante a expresiones faciales. *Tesis CENIDET*, 1 of 122.

Mallick, S. (2015, October 30). *OpenCV (C ++ vs Python) vs MATLAB para*

Computer Vision | LearnOpenCV #. https://learnopencv.com/opencv-c-vs-python-vs-matlab-for-computer-vision/?fbclid=IwAR2DhY_2lHLPTyz4fvKZmM-J25WQunEBsdeN1W3AObjhDBjjgyw4apHIWXw

Martinez, A, & Benavente, R. (1998). The AR face database, CVC. *Copyright of Informatica (03505596)*.

Martinez, Aleix, & Benavente, R. (1998). *The AR Face Database: CVC Technical Report, 24*.

MediaPipe. (2020). *Face Detection - mediapipe*. https://google.github.io/mediapipe/solutions/face_detection

Mitchell, T. M., & others. (1997). *Machine learning*.

Mollahosseini, A., Hasani, B., & Mahoor, M. H. (2017). Affectnet: A database for facial expression, valence, and arousal computing in the wild. *IEEE Transactions on Affective Computing*, *10*(1), 18–31.

Montero, D., Nieto, M., Leskovsky, P., & Aginako, N. (2021). Boosting Masked Face Recognition with Multi-Task ArcFace. *ArXiv Preprint ArXiv:2104.09874*.

Mucha, J. E., Ortega, A. R., Salazar, L. M. R., Montero, M. M. A., Roy, C., Pahuacho, U., Moreno, A. E. G., Nacional, U., & Castilla, M. (2021). Aplicación del deep learning para el reconocimiento facial con la presencia de oclusiones en el contexto de la pandemia covid 2021. *Revista ECIPeru*, 17–24. https://doi.org/10.33017/reveciperu2021.0002/

Mundial, I. Q., Ul Hassan, M. S., Tiwana, M. I., Qureshi, W. S., & Alanazi, E. (2020). Towards facial recognition problem in COVID-19 pandemic. *2020 4th International Conference on Electrical, Telecommunication and Computer Engineering, ELTICOM 2020 - Proceedings*, 210–214. https://doi.org/10.1109/ELTICOM50775.2020.9230504

Myler, H. R., & Weeks, A. R. (1993). *The pocket handbook of imaging processing algorithms in C*. Prentice-Hall, Inc.

Nabatchian, A., Abdel-Raheem, E., & Ahmadi, M. (2011). Illumination invariant feature extraction and mutual-information-based local matching for face recognition under illumination variation and occlusion. *Pattern Recognition*, *44*(10–11), 2576–2587.

Nagrath, P., Jain, R., Madan, A., Arora, R., Kataria, P., & Hemanth, J. (2021). SSDMNV2: A real time DNN-based face mask detection system using single shot multibox detector and MobileNetV2. *Sustainable Cities and Society*, *66*, 102692.

Naqa, I. El, & Murphy, M. J. (2015). Machine Learning in Radiation Oncology. *Machine Learning in Radiation Oncology*, 3–11. https://doi.org/10.1007/978-3-319-18305-3

Nuñez, A. V., & Nuñez, L. N. (2020). Face recognition for automatic vehicle ignition based on Raspberry Pi. *Revista Odigos*, *1*(2), 53–68. https://doi.org/10.35290/ro.v1n2.2020.326

Olivares-Mercado, J., Toscano-Medina, K., Sánchez-Pérez, G., Nakano-Miyatake, M., & Pérez-Meana, H. (2016). Modifications to the Eigenphases Method for Face Recognition Based on SVM. *Ingenier{\'\i}a, Investigación y Tecnolog{\'\i}a*, *17*(1), 119–129.

ONU. (2020, December 1). *Cuándo y cómo usar mascarilla*. https://www.who.int/es/emergencies/diseases/novel-coronavirus-2019/advice-for-public/when-and-how-to-use-masks?gclid=CjwKCAjwuvmHBhAxEiwAWAYj-FbUTLtPvq0dhraHhdvZeY12GFNqk0DNM0pVCezcgP69pY7_4As4-RoCYioQAvD_BwE

Organization, W. H., & others. (2020). *Advice on the use of masks in the context of COVID-19: interim guidance, 5 June 2020*.

Parker, J. R. (2010). *Algorithms for image processing and computer vision*. John Wiley \& Sons.

Parkhi, O. M., Vedaldi, A., & Zisserman, A. (2015). *Deep face recognition*.

Pedregosa, F., Varoquaux, G., Gramfort, A., & Michel, V. (2011a). *1.11. Ensemble methods — scikit-learn 1.1.1 documentation*. https://scikit-learn.org/stable/modules/ensemble.html#forest

Pedregosa, F., Varoquaux, G., Gramfort, A., & Michel, V. (2011b). *1.4. Máquinas de vectores de soporte — documentación de scikit-learn - 1.1.1*. https://scikit-learn.org/stable/modules/svm.html#classification

prajnasb. (2020, April 14). *GitHub - prajnasb/observations*. https://github.com/prajnasb/observations

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2015). You only look once: Unified, real-time object detection. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 779–788.

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *ArXiv Preprint ArXiv:1506.01497*.

Rosebrock, A. (2015, December 7). *Patrones binarios locales con Python y OpenCV - PyImageSearch*. https://pyimagesearch.com/2015/12/07/local-binary-patterns-with-python-opencv/

Rosebrock, A. (2017, April 3). *Facial landmarks with dlib, OpenCV, and Python - PyImageSearch*. https://pyimagesearch.com/2017/04/03/facial-landmarks-dlib-opencv-python/

Rosebrock, A. (2021, April 19). *Detección de rostros con dlib (HOG y CNN) - PyImageSearch*. https://pyimagesearch.com/2021/04/19/face-detection-with-dlib-hog-and-cnn/

Samuel, A. L. (1959). Eight-move opening utilizing generalization learning. (See Appendix B, Game G-43.1 Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal*, 210–229.

Sharif, M., Mohsin, S., Hanan, R. A., Javed, M. Y., & Raza, M. (2011). Using nose

heuristics for efficient face recognition. *Sindh University Research Journal-SURJ (Science Series)*, *43*(1 (a)).

Shipman, J. W. (2013). Tkinter 8.5 reference: a GUI for Python. *Computer*, 1–118. tcc-doc@nmt.edu

Simonyan, K., & Zisserman, A. (2014). Very deep convolutional networks for large-scale image recognition. *ArXiv Preprint ArXiv:1409.1556*.

Singh, S., & Prasad, S. V. A. V. (2018). Techniques and challenges of face recognition: A critical review. *Procedia Computer Science*, *143*, 536–543. https://doi.org/10.1016/j.procs.2018.10.427

sixphere, Luque, C., & Jurado, J. (2020, May 12). *Detección automática de temperatura y uso de mascarilla*. https://sixphere.com/laboratorio-deteccion-temperatura-mascarilla/#entrevista-jesus

Smola, A. J., & Schölkopf, B. (2004). A tutorial on support vector regression. *Statistics and Computing*, *14*(3), 199–222.

Song, L., Gong, Di., Li, Z., Liu, C., & Liu, W. (2019). Occlusion robust face recognition based on mask learning with pairwise differential siamese network. *Proceedings of the IEEE International Conference on Computer Vision*, *2019-Octob*, 773–782. https://doi.org/10.1109/ICCV.2019.00086

Soto, D. A. P. (2009). Seguimiento y Caracterización de Componentes del Rostro para la Detección de Expresiones Faciales. *Tesis CENIDET*.

Uijlings, J. R. R., Van De Sande, K. E. A., Gevers, T., & Smeulders, A. W. M. (2013). Selective search for object recognition. *International Journal of Computer Vision*, *104*(2), 154–171.

Valderrama, W. C. (2019). Reconocimiento automático del rostro para verificación de identidad para evaluación en línea. *Tesis CENIDET*, 102.

Vázquez, C. A. (2017). Valoración Automática de la motivación y atención del estudiante mediante expresiones faciales. *Tesis CENIDET*.

Wan, W., & Chen, J. (2018). Occlusion robust face recognition based on mask learning. *Proceedings - International Conference on Image Processing, ICIP*, *2017-Septe*, 3795–3799. https://doi.org/10.1109/ICIP.2017.8296992

Wang, Z., Wang, G., Huang, B., Xiong, Z., Hong, Q., Wu, H., Yi, P., Jiang, K., Wang, N., Pei, Y., Chen, H., Miao, Y., Huang, Z., & Liang, J. (2020). Masked face recognition dataset and application. *ArXiv*, 1–3.

Wazwaz, A. A., Herbawi, A. O., Teeti, M. J., & Hmeed, S. Y. (2018). Raspberry Pi and computers-based face detection and recognition system. *2018 4th International Conference on Computer and Technology Applications, ICCTA 2018*, 171–174. https://doi.org/10.1109/CATA.2018.8398677

Wu, X., Zhou, J., & Pan, Y. (2017). Initial Shape Pool Construction for Facial Landmark Localization under Occlusion. *IEEE Access*, *5*, 16649–16655. https://doi.org/10.1109/ACCESS.2017.2739822

Wudan, Y. (2020, September 14). *¿Llevas puesta la mascarilla? Un software de*

reconocimiento está listo para chequear si las personas cumplen con el correcto uso | National Geographic. https://www.nationalgeographicla.com/ciencia/2020/09/software-reconocimiento-mascarillas

Yang, S., Luo, P., Loy, C.-C., & Tang, X. (2016). Wider face: A face detection benchmark. *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 5525–5533.

Yi, D., Lei, Z., Liao, S., & Li, S. Z. (2014). Learning face representation from scratch. *ArXiv Preprint ArXiv:1411.7923*.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016a). Joint face detection and alignment using multitask cascaded convolutional networks. *IEEE Signal Processing Letters*, *23*(10), 1499–1503.

Zhang, K., Zhang, Z., Li, Z., & Qiao, Y. (2016b). Joint Face Detection and Alignment Using Multitask Cascaded Convolutional Networks. *IEEE Signal Processing Letters*, *23*(10), 1499–1503. https://doi.org/10.1109/LSP.2016.2603342

Zhang, T., Li, J., Jia, W., Sun, J., & Yang, H. (2018). Fast and robust occluded face detection in ATM surveillance. *Pattern Recognition Letters*, *107*, 33–40. https://doi.org/10.1016/j.patrec.2017.09.011

Zhao, S. (2018). Pixel-level occlusion detection based on sparse representation for face recognition. *Optik*, *168*(May), 920–930. https://doi.org/10.1016/j.ijleo.2018.05.013

Zitnick, C. L., & Dollár, P. (2014). Edge boxes: Locating object proposals from edges. *European Conference on Computer Vision*, 391–405.

Ziwei, X., Liang, Z., Jingyu, P., Jinqian, Z., Hongling, C., Yiwen, Z., Xi, H., Siyuan, X., & Haoyang, Y. (2020). Face Occlusion Detection Based on SSD Algorithm. *ICEIEC 2020 - Proceedings of 2020 IEEE 10th International Conference on Electronics Information and Emergency Communication*, 362–365. https://doi.org/10.1109/ICEIEC49280.2020.9152335

Anexos

anexo A

Selección del Ambiente de Sistema

Se selecciono trabajar con Python como lenguaje de programación para desarrollar el sistema gracias a las múltiples herramientas y librerías que se pueden integrar a él como es OpenCV, además de que no requiere gran espacio en disco con lo cual se podría implementar en una Raspberry Pi si así fuese necesario.

Ventajas de Python con OpenCV (Mallick, 2015).

- **Facilidad de uso**: Si eres un programador de Python, usar OpenCV (Python) sería muy fácil. Python es un lenguaje fácil de aprender (especialmente en comparación con C ++).

- **Python se ha convertido en el lenguaje de la computación científica**: Hace unos años, MATLAB fue llamado el lenguaje de la computación científica. Pero ahora, con OpenCV, numpy, scipy, scikit-learn y matplotlib, Python proporciona un entorno poderoso para aprender y experimentar con la visión por computadora y el aprendizaje automático.

- **Visualización y depuración:** cuando usa OpenCV (Python), tiene acceso a una gran cantidad de bibliotecas escritas para Python. La visualización con matplotlib es tan buena como MATLAB. El código de depuración en Python es más fácil que en C ++, pero no coincide con la súper facilidad de MATLAB.

- **Creación de backend web**: Python también es un lenguaje popular para la creación de sitios web. Es muy fácil usar OpenCV (Python) junto con estos marcos web.

Nota: En lo personal al momento de buscar información relacionada con OpenCV siempre encuentro más fácilmente información que se relaciona con Python en comparación con C++.

Diseño del Sistema

Para facilidad del usuario se creó una pequeña interfaz con ayuda de la biblioteca *Tkinter 8.5* (Shipman, 2013) que se tiene instalada por defecto al instalar Python. La interfaz se integra de 6 botones de operación para el usuario, como se muestra en la figura B1.

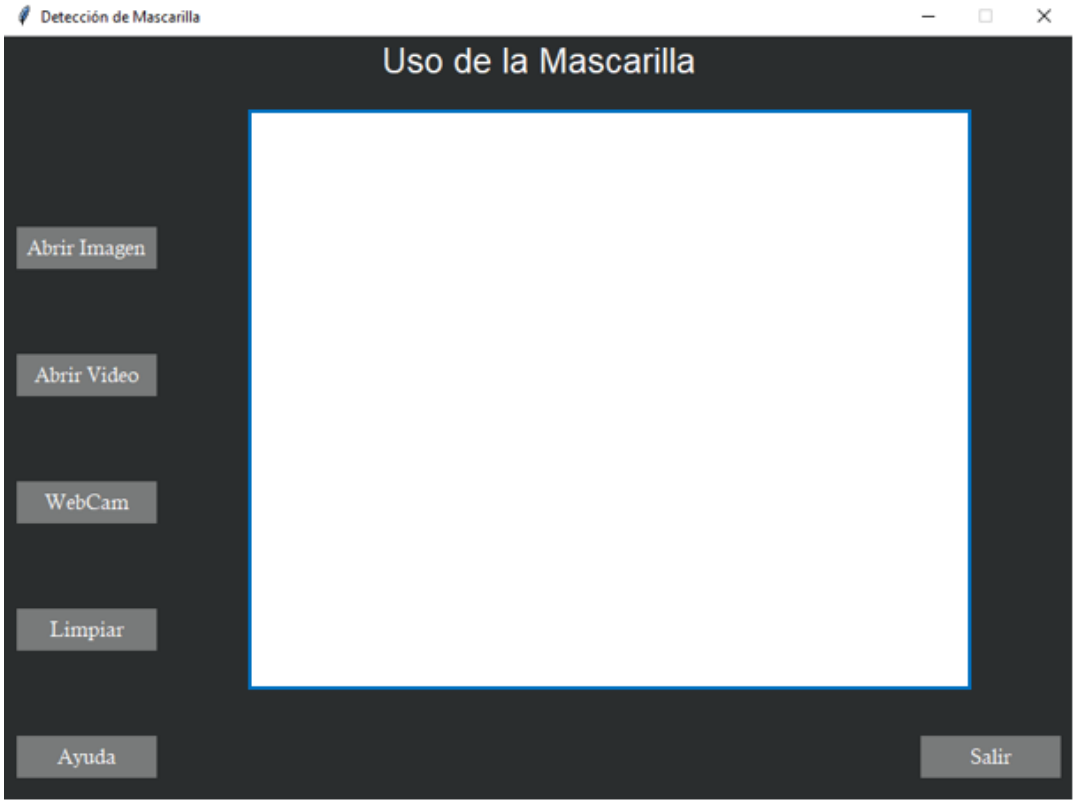


Figura B1 Vista principal de la Interfaz.

El sistema permite el ingreso de las imágenes de tres formas: a partir de una imagen individual, de un video y a través de la cámara Web predefinida de la computadora donde se está ejecutando, para tener el control de cada modalidad se asignó un botón correspondiente al nombre de la función que realiza.

Al presionar el botón de “Abrir imagen” se muestra la ventana que permite buscar en la computadora la imagen con que se desea trabajar (en formato .jpeg, .jpg y .png) como se puede apreciar en la figura B2.

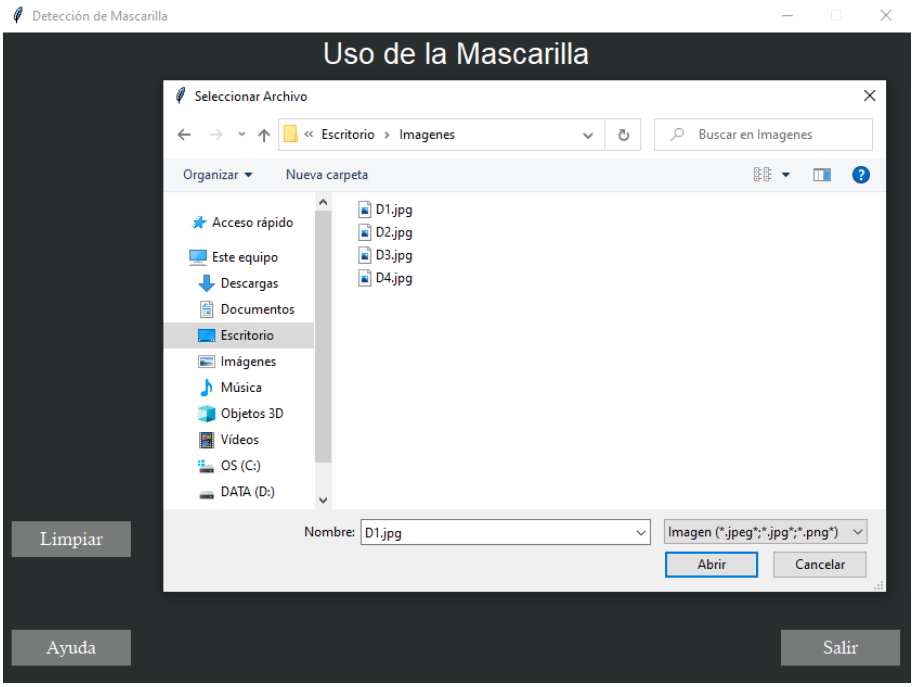


Figura B2 Vista de selección de una imagen a procesar.

En el caso de presionar el botón de “Abrir video” aparece la ventana que permite buscar en la computadora el video, en formato .mp4 o .avi, con que se desea trabajar como se puede apreciar en la figura B3.

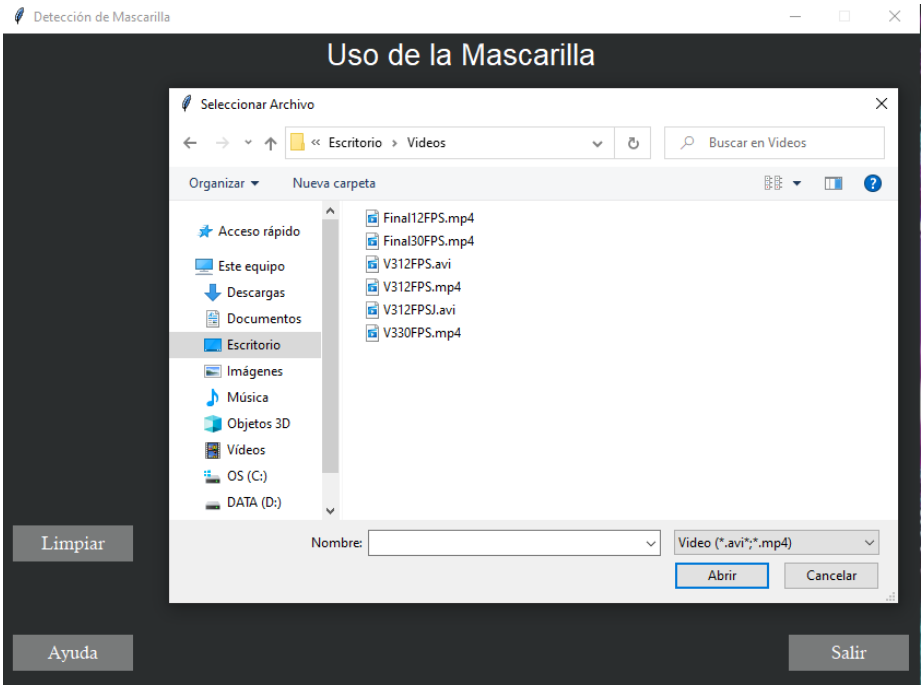


Figura B3 Vista de selección de un video a procesar.

En el caso de presionar el botón de “WebCam” se habilita la cámara por defecto de la computadora donde se está ejecutando el sistema, como se puede apreciar en la figura B4.

Figura B4 Vista de la Webcam con la imagen a procesar.

Cuando se pulsa una de las tres modalidades mencionadas, automáticamente, se desactivan las demás. También se activa el botón de “Limpiar” el cual señala el término del proceso elegido antes (imagen de entrada por video, imagen o webcam) y regresa al estado inicial, donde de nuevo se puede seleccionar nuevamente la modalidad deseada. En el caso de ya no querer continuar, se pulsa el botón de “Salir” con el cual se cerrará el programa y detendrá la actividad que se esté ejecutando. Con cada modalidad se guarda la imagen o el video procesado con la fecha y hora en que se analizó.

También se cuenta con un botón de “Ayuda” en el cual se describe la metodología final que se aplica a cada imagen o fotograma del video, como se puede observar en la figura B5.

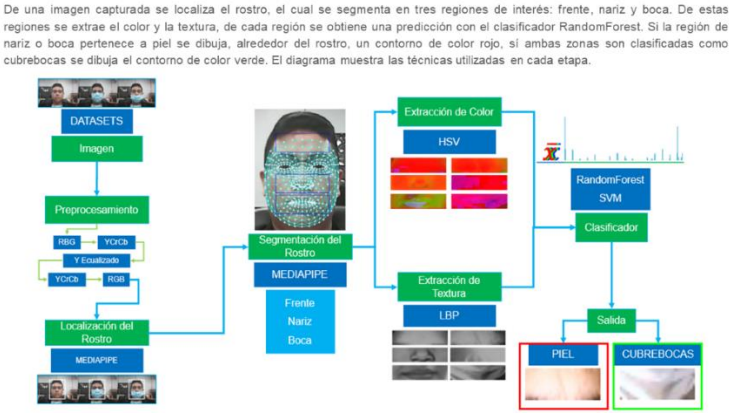


Figura B5 Visualización de ayuda.

anexo C

Herramientas Utilizadas y Características de la Computadora

La interfaz se creó mediante la librería *Tkinter 8.5* (Shipman, 2013), el sistema utilizando *Python* 3.9.6[17]. Para trabajar con videos e imágenes se utilizaron las librerías de *Opencv-contrib-python* 4.5.5.64[18] mismas que permiten guardar la imagen o video para, posteriormente, ser revisados.

Para facilitar la visualización en la interfaz se ocupa *Imutils 0.5.4*[19] que redimensiona la imagen de entrada a una altura de 480 y obteniendo la proporción correspondiente para la amplitud, de tal forma que se mantenga dentro de las dimensiones de la interfaz. En el caso de los videos (tanto los almacenados en la computadora o los adquiridos por la webcam) para poder visualizarlos se utiliza la librería *Pillow* 8.3.2[20] para que no se quede congelada la imagen en la ventana de la interfaz.

Para la etapa de detección y segmentación del rostro en las tres regiones de interés se utiliza *MediaPipe* 0.8.9.1[21]. Se utilizan las imágenes de las tres regiones para la extracción de características con LBP utilizando la librería *Scikit-image* 0.19.2[22]. Para la conversión del espacio de color se usa *Opencv-contrib-python* 4.5.5.64[23]. El vector de características se construye utilizando la librería de *Numpy* 1.21.2[24]. Y finalmente, con ayuda de *Joblib* 1.0.1[25] se carga el mejor modelo de clasificación, obtenido de la etapa de entrenamiento, y que se utiliza al final en el sistema.

Tanto la interfaz como el sistema se desarrollaron en una laptop Asus X515JA con Windows 10 versión 21H1 64 bits, con procesador Intel(R) Core (TM) i3-1005G1 CPU @ 1.20GHz, disco de estado sólido/EMMC1 256GB, memoria RAM 12GB.

A continuación, se muestran los comandos en el símbolo del sistema en Windows para la instalación de las librerías necesarias para el funcionamiento del sistema en caso de no contar con el ejecutable.

[17] https://www.python.org/downloads/
[18] https://pypi.org/project/opencv-contrib-python/
[19] https://pypi.org/project/imutils/
[20] https://pypi.org/project/Pillow/8.3.2/
[21] https://pypi.org/project/mediapipe/
[22] https://pypi.org/project/scikit-image/
[23] https://pypi.org/project/opencv-contrib-python/
[24] https://pypi.org/project/numpy/
[25] https://pypi.org/project/joblib/

En caso de contar con la versión 2.7 de Python además de la versión 3.9 sustituya el comando pip por pip3 como se muestra a continuación:

```
pip3 install mediapipe==0.8.11
```

Verificar la versión de python

```
python –version
```

Mediapipe 0.8.9.1

```
pip install mediapipe==0.8.9.1
```

```
pip3 install mediapipe==0.8.9.1
```

opencv-contrib-python 4.5.5.64

```
pip install opencv-contrib-python==4.5.5.64
```

```
pip3 install opencv-contrib-python==4.5.5.64
```

imutils 0.5.4

```
pip install imutils==0.5.4
```

```
pip3 install imutils==0.5.4
```

pillow 8.3.2

```
pip install Pillow==8.3.2
```

```
pip3 install Pillow==8.3.2
```

scikit-image 0.19.2

```
pip install scikit-image==0.19.2
```

```
pip3 install scikit-image==0.19.2
```

numpy 1.21.2

```
pip install numpy==1.21.2
```

```
pip3 install numpy==1.21.2
```

joblib 1.0.1

```
pip install joblib==1.0.1
```

```
pip3 install joblib==1.0.1
```

A continuación, se muestran las versiones más actuales con las que se encontró que el sistema podía trabajar correctamente.

Mediapipe 0.8.11

pip install mediapipe==0.8.11

pip3 install mediapipe==0.8.11

opencv-contrib-python 4.6.0.66

pip install opencv-contrib-python==4.6.0.66

pip3 install opencv-contrib-python==4.6.0.66

imutils 0.5.4

pip install imutils==0.5.4

pip3 install imutils==0.5.4

pillow 9.2.0

pip install Pillow==9.2.0

pip3 install Pillow==9.2.0

scikit-image 0.19.3

pip install scikit-image==0.19.3

pip3 install scikit-image==0.19.3

scikit-learn 1.1.2

pip install scikit-learn ==1.1.2

pip3 install scikit-learn ==1.1.2

sklearn 0.0

pip install sklearn ==0.0

pip3 install sklearn ==0.0

numpy 1.23.4

pip install numpy==1.23.4

pip3 install numpy==1.23.4

joblib 1.2.0

pip install joblib==1.2.0

pip3 install joblib==1.2.0

Código de la Implementación

```
from skimage.feature import local_binary_pattern
from PIL import Image, ImageTk
from tkinter import filedialog
from datetime import datetime
from tkinter import Label
from tkinter import Toplevel
import mediapipe as mp
import tkinter as tk
import numpy as np
import warnings
import imutils
import joblib
import time
import cv2
import os
#----------------------------------------------------------
warnings.filterwarnings('ignore', '.*do not.*',)
def resource_path(relative_path):
    try:
        base_path = sys._MEIPASS
    except Exception:
        base_path = os.path.abspath(".")
    return os.path.join(base_path, relative_path)
def Ecualizado(img):
    ycrcb=cv2.cvtColor(img, cv2.COLOR_BGR2YCrCb)
    (y, cr, cb) = cv2.split(ycrcb)
    y = cv2.equalizeHist(y)
    merged = cv2.merge([y, cr, cb])
    resultado= cv2.cvtColor(merged, cv2.COLOR_YCrCb2BGR)
    return resultado
def imagenpuntos(img,P_cx,P_cy,r1,r2,op):
    if(op==1):
        x = int(P_cx)
        y = int(P_cy)
        #r1 largo de la imagen
        #r2 alto de la imagen
        rectX =int(x - r1)
        rectY =int(y - r2)
        if(rectX < 0):
            rectX = 0
```

```
        img1 = img[rectY:(y),rectX:(x+r1)]
    if(op==2):
        x = int(P_cx)
        y = int(P_cy)
        r2= 6
        rectX =int(x - r1)
        rectY =int(y - r2)
        if(rectX < 0):
            rectX = 0
        img1 = img[rectY:(y+r2),rectX:(x+r1)]
    if img1 is None:
        img2=np.zeros((150,50,3),np.uint8)
    else:
        img2 = cv2.resize(img1,(150,50))
    return img2
def promediohsv(img):
    height = img.shape[0]
    width = img.shape[1]
    dimension=0
    dimension=height*width
    suma1=0
    for i in range(height):
        for j in range(width):
          suma1+=img[i][j]
    Prom1=0
    Prom1=suma1/dimension
    return round(Prom1)
def promedioimg1(img):
    img_HSV = cv2.cvtColor(img, cv2.COLOR_BGR2HSV)
    (HH, SS, VV) = cv2.split(img_HSV)
    prom1=promediohsv(HH)
    prom2=promediohsv(SS)
    prom3=promediohsv(VV)
    PromF=0
    PromF=(prom1+prom2+prom3)/3
    return round(PromF)

def calcHisto(img):
    hist = cv2.calcHist(img, [0], None, [256], [0, 256])
    cv2.normalize(hist,hist,0,255,cv2.NORM_MINMAX)
    hist=np.int32(np.around(hist))
    return hist
def arreglo258(PromedioColor,HistoTextura,Clase):
    HistoTextura = np.insert(HistoTextura,0,PromedioColor,axis=None)
    HistoTextura = np.append(HistoTextura,Clase,axis=None)
    return HistoTextura
def tabla_de_verdad(PrediF,PrediN,PrediB):
```

```
    if PrediN==2 and PrediB==2:
        color=(0,255,0)#V
        bg="#1DB954"
        textmask="Cubrebocas Usado Correctamente"
    else:
        color=(255,0,0)#R
        bg="#FF0000"
        textmask="Use el Cubrebocas"
    return color,bg,textmask
def claseModelo(img,PromedioColor,Modelo1):
    grayF = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
    lbpF = local_binary_pattern(grayF, 8, 1)
    HistoLBPF=calcHisto(lbpF.astype("uint8"))
    HistoTextura = np.insert(HistoLBPF,0,PromedioColor,axis=None)
    HistoTextura=HistoTextura.reshape(1, -1)
    Prediccion=Modelo1.predict(HistoTextura)
    return int(Prediccion)
mp_face_mesh = mp.solutions.face_mesh
LEFT_IRIS = [474,475, 476, 477]
RIGHT_IRIS = [469, 470, 471, 472]
def eyes_detect(imgeyes,Modelo1):
    with mp_face_mesh.FaceMesh(
    max_num_faces=1,refine_landmarks=True,
    min_detection_confidence=0.5,min_tracking_confidence=0.5) as face_mesh:
        rgb_frame = cv2.cvtColor(imgeyes, cv2.COLOR_BGR2RGB)
        img_h, img_w = rgb_frame.shape[:2]
        results = face_mesh.process(rgb_frame)
        if results.multi_face_landmarks:
            mesh_points=np.array([np.multiply([p.x, p.y], [img_w, img_h]).astype(int)
            for p in results.multi_face_landmarks[0].landmark])
            (l_cx, l_cy), l_radius = cv2.minEnclosingCircle(mesh_points[LEFT_IRIS])
            (r_cx, r_cy), r_radius = cv2.minEnclosingCircle(mesh_points[RIGHT_IRIS])
            (FT_cx,                                                    FT_cy),radioFT=
cv2.minEnclosingCircle(mesh_points[[8,10]])#Puntos de Frente Temporal
            (F_cx,                                                  F_cy),radiosinusoF=
cv2.minEnclosingCircle(mesh_points[[9]])#Puntos de Frente
            (P_cx,                                                    P_cy),radioojos=
cv2.minEnclosingCircle(mesh_points[[33,263]])#Puntos de ojo 133,362
            (NT_cx,                                                  NT_cy),radioNT=
cv2.minEnclosingCircle(mesh_points[[164,195]])#Puntos de Nariz Temporal
            (N_cx,                                                  N_cy),radiosinuso1n=
cv2.minEnclosingCircle(mesh_points[[4]])#Puntos de Nariz
            (BT_cx,                                                  BT_cy),radioBT=
cv2.minEnclosingCircle(mesh_points[[164,199,428]])#Puntos de Boca Temporal
            (B_cx,                                                  B_cy),radiosinuso1B=
cv2.minEnclosingCircle(mesh_points[[13,14]])#Puntos de Boca
            mediox=(l_cx+r_cx)/2
```

```
            center_left = np.array([l_cx, l_cy], dtype=np.int32)
            center_right = np.array([r_cx, r_cy], dtype=np.int32)
            frente = np.array([F_cx, F_cy], dtype=np.int32)
            nariz1 = np.array([N_cx, N_cy], dtype=np.int32)
            Boca1 = np.array([B_cx, B_cy], dtype=np.int32)
            frente= imagenpuntos(imgeyes,F_cx,F_cy,int(radioojos),int(radioFT),1)
            nariz= imagenpuntos(imgeyes,N_cx,N_cy,int(radioojos),int(radioNT),2)
            boca= imagenpuntos(imgeyes,B_cx,B_cy,int(radioojos),int(radioBT),2)
            curDT = datetime.now()
            date_time = curDT.strftime("%m/%d/%Y, %H:%M:%S")
            Prom1=promedioimg1(frente)
            Prom2=promedioimg1(nariz)
            Prom3=promedioimg1(boca)
            claseF=claseModelo(frente,Prom1,Modelo1)
            claseN=claseModelo(nariz,Prom2,Modelo1)
            claseB=claseModelo(boca,Prom3,Modelo1)
            colorprom,bg,textmask=tabla_de_verdad(claseF,claseN,claseB)
        else:
            frente=imgeyes
            nariz=imgeyes
            boca=imgeyes
            colorprom=(255,255,255)
            bg='#FFFFFF'#Azul Aqua
            textmask="Uso Correcto del Cubrebocas"
        return colorprom,bg,textmask,imgeyes
mp_face_detection = mp.solutions.face_detection
mp_drawing = mp.solutions.drawing_utils
def Mediapipe_face(img3M,img1,color1,Modelo):
    with      mp_face_detection.FaceDetection(min_detection_confidence=0.5)      as
face_detection:
        resultados = face_detection.process(img3M)
        height, width, _ = img3M.shape
        if resultados.detections is not None:
            for detection in resultados.detections:
                xmin = int(detection.location_data.relative_bounding_box.xmin * width)
                ymin = int(detection.location_data.relative_bounding_box.ymin * height)
                w = int(detection.location_data.relative_bounding_box.width * width)
                h = int(detection.location_data.relative_bounding_box.height * height)
            img3M1=img3M.copy()
            detecccion=img3M1[ymin-25:ymin+h,xmin:xmin+w]
            if ymin-25>=0:
                if detecccion is None:
                    detecccion1=np.zeros((128,128,3),np.uint8)
                else:
                    detecccion1 = cv2.resize(detecccion,(128,128))
                color1,bg,textmask,img3M1=eyes_detect(detecccion1,Modelo)
                img1=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
```

```
                cv2.rectangle(img1, (xmin, ymin-25), (xmin + w, ymin + h), color1, 2)
            else:
                bg='#FFFFFF'#Azul Aqua
                textmask="Uso del Cubrebocas"
                img1=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
        else:
            bg='#FFFFFF'#Azul Aqua
            textmask="Uso del Cubrebocas"
            img1=cv2.cvtColor(img1,cv2.COLOR_BGR2RGB)
        return img1,bg,textmask
pathModel=resource_path("Modelo.pkl")
Modelo1=joblib.load(pathModel)
bg1="#2a2d2e"
bg2="#787a7a"
ventana=tk.Tk()
ventana.geometry("840x600")
ventana.title("Detección de Mascarilla")
ventana.resizable(width=False,height=False)
pathIco=resource_path("icon.ico")
ventana.iconbitmap(pathIco)
ventana.configure(bg=bg1)
label=Label(ventana,text = "Uso de la Mascarilla",bg=bg1,fg="white",font=("Arial",
20))
label.pack()
def close_window ():
    global dato
    if dato==1:
        video.release()
    ventana.destroy()
def desactiva():
    boton2['state'] = tk.DISABLED
    boton2.place_forget()
    boton3['state'] = tk.DISABLED
    boton3.place_forget()
    boton4['state'] = tk.DISABLED
    boton4.place_forget()
    boton5['state'] = tk.NORMAL
    boton5.place(x=10,y=450)
def abrir_imagen():
    global ventana2
    global video
    global dato
    dato=0
    ventana2=tk.Label(ventana,bg="white")
    desactiva()
    filename = filedialog.askopenfilename(initialdir = "/Desktop/",title = "Seleccionar
Archivo",filetypes = (("Imagen","*.jpeg*"),("Imagen","*.jpg*"),("Imagen","*.png*")))
```

```
    frame=cv2.imread(filename,1)
    if frame is None:
        Limpiar()
    else:
        frame = imutils.resize(frame, height = 480)
        frame = cv2.flip(frame, 1)
        frame1=frame.copy()
        frameE=Ecualizado(frame)
        frame,bgcolor,textmask=Mediapipe_face(frameE,frame1,(255,                255,
255),Modelo1)
        ventana.configure(bg=bgcolor)
        label.config(text=textmask,bg=bgcolor)
        ventana2.place(x=150,y=50)
        framesave=cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        img=Image.fromarray(frame)
        image=ImageTk.PhotoImage(image=img)
        ventana2.configure(image=image)
        ventana2.image=image
        now = datetime.now()
        current_time = now.strftime("%d_%m_%Y_%H_%M_%S")
        nameimg="Img"+current_time+".jpg"
        cv2.imwrite(nameimg, framesave)
def savevideo():
    global out
    now = datetime.now()
    current_time = now.strftime("%d_%m_%Y_%H_%M_%S")
    namevideosave="Video"+current_time+".avi"
    out = cv2.VideoWriter(namevideosave, cv2.VideoWriter_fourcc(*'XVID'), 30,(640,
480))
def video_Read():
    desactiva()
    global video
    global dato
    dato=1
    filename = filedialog.askopenfilename(initialdir = "/Desktop/",title = "Seleccionar
Archivo",filetypes = (("Video","*.avi*"),("Video",".mp4")))
    video=cv2.VideoCapture(filename)
    savevideo()
    abrir_video()
def abrir_video():
    global ventana2
    global dato
    global out
    dato=1
    boton5['state'] = tk.NORMAL
    boton5.place(x=10,y=450)
    global video
```

```
    desactiva()
    ret,frame=video.read()
    frame = cv2.flip(frame, 1)
    if ret==True:
        ventana2.place(x=150,y=50)
        frame = cv2.resize(frame,(640,480))
        frame1=frame.copy()
        frameE=Ecualizado(frame)
        frame,bgcolor,textmask=Mediapipe_face(frameE,frame1,(255,          255,
255),Modelo1)
        ventana.configure(bg=bgcolor)
        label.config(text=textmask,bg=bgcolor)
        framesave=cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        out.write(framesave)
        img=Image.fromarray(frame)
        image=ImageTk.PhotoImage(image=img)
        ventana2.configure(image=image)
        ventana2.image=image
        ventana2.after(5,abrir_video)
    else:
        Limpiar()
def video_stream():
    desactiva()
    global video
    global dato
    dato=1
    video=cv2.VideoCapture(0)
    savevideo()
    WebCam()
#video es 640x480
def WebCam():
    boton5['state'] = tk.NORMAL
    boton5.place(x=10,y=450)
    global video
    global dato
    global out
    dato=1
    desactiva()
    ret,frame=video.read()
    frame = cv2.flip(frame, 1)
    if ret==True:
        ventana2.place(x=150,y=50)
        frame = cv2.resize(frame,(640,480))
        frame1=frame.copy()
        frameE=Ecualizado(frame)
        frame,bgcolor,textmask=Mediapipe_face(frameE,frame1,(255,          255,
255),Modelo1)
```

```
        ventana.configure(bg=bgcolor)
        label.config(text=textmask,bg=bgcolor)
        framesave=cv2.cvtColor(frame,cv2.COLOR_BGR2RGB)
        out.write(framesave)
        img=Image.fromarray(frame)
        image=ImageTk.PhotoImage(image=img)
        ventana2.configure(image=image)
        ventana2.image=image
        ventana2.after(5,WebCam)
    else:
        Limpiar()
global dato
dato=0
def Limpiar():
    global video
    global dato
    boton2['state'] = tk.NORMAL
    boton2.place(x=10,y=150)
    boton3['state'] = tk.NORMAL
    boton3.place(x=10,y=250)
    boton4['state'] = tk.NORMAL
    boton4.place(x=10,y=350)
    boton5['state'] = tk.DISABLED
    boton5.place_forget()
    ventana2.place_forget()
    ventana.configure(bg=bg1)
    label.config(text="Uso de la Mascarilla",bg=bg1)
    if dato==1:
        video.release()
        out.release()
def ayuda():
    windowAyuda = Toplevel(ventana)
    windowAyuda.title("Ayuda")
    windowAyuda.geometry("1000x563")
    windowAyuda.resizable(width=False,height=False)
    pathAyuda=resource_path("metodologia.png")
    fondo=tk.PhotoImage(file=pathAyuda)

fondo1=tk.Label(windowAyuda,image=fondo).place(x=0,y=0,relwidth=1,relheight=1
)
    windowAyuda.image=fondo
boton1                                                                    =
tk.Button(ventana,text="Salir",bg=bg2,fg="white",relief="flat",cursor="hand2",comm
and=close_window,width=10,height=1, font=("Calisto MT",12,"normal"))
boton2                          =                    tk.Button(ventana,text="Abrir
Imagen",bg=bg2,fg="white",relief="flat",cursor="hand2",command=abrir_imagen,wi
dth=10,height=1, font=("Calisto MT",12,"normal"))
```

```
boton3 = tk.Button(ventana,text="Abrir Video",bg=bg2,fg="white",relief="flat",cursor="hand2",command=video_Read,width=10,height=1, font=("Calisto MT",12,"normal"))
boton4 = tk.Button(ventana,text="WebCam",bg=bg2,fg="white",relief="flat",cursor="hand2",command=video_stream,width=10,height=1, font=("Calisto MT",12,"normal"))
boton5 = tk.Button(ventana,text="Limpiar",bg=bg2,fg="white",relief="flat",cursor="hand2",command=Limpiar,width=10,height=1, font=("Calisto MT",12,"normal"))
boton6 = tk.Button(ventana,text="Ayuda",bg=bg2,fg="white",relief="flat",cursor="hand2",command=ayuda,width=10,height=1, font=("Calisto MT",12,"normal"))
boton1.place(x=720,y=550)
boton2.place(x=10,y=150)
boton3.place(x=10,y=250)
boton4.place(x=10,y=350)
boton5.place(x=10,y=450)
boton5.place_forget()
boton5['state'] = tk.DISABLED
boton6.place(x=10,y=550)
ventana2=tk.Label(ventana,bg=bg1)
ventana2.place(x=150,y=50)
ventana.mainloop()
```