ISSN 1870-4069

Metodología para la detección de mascarilla mediante aprendizaje automático

Ma. del Carmen Santiago, Ana C. Zenteno, Yeiny Romero,
Judith Pérez, Gustavo T. Rubín, Antonio Álvarez, Alexis Meza

Benemérita Universidad Autónoma de Puebla,
Facultad de Ciencias de la Computación,
México

{marycarmen.santiago, ana.zenteno, yeiny.romero, judith.perez,
gustavo.rubin}@correo.buap.mx, eduard-alvarez@live.com.mx,
alexis.meza@alumno.buap.mx

Resumen. El Covid-19 nos ha obligado a emprender acciones de protección para evitar el contagio, una de éstas es el uso de mascarillas. El reconocimiento facial (RF) tiene sus bases desde 1883, gracias a su estudio se tienen técnicas que se basan en modelos y apariencias. El RF requiere de tres etapas: a) Detección del rostro en una imagen, b) extracción de las características y c) identificación y/o verificación. En este trabajo se presenta la metodología para llevar a cabo a través del RF, la detección de la mascarilla en el rostro. Se presentan los primeros resultados favorables a partir de los cuales se proponen las bases para un proceso de optimización que busca realizar la identificación del uso correcto de la mascarilla.

Palabras clave: COVID-19, reconocimiento facial, detección mascarilla.

Methodology for Mask Detection Using Machine Learning

Abstract. Covid-19 has forced us to take protective actions to avoid contagion, one of these is the use of masks. Facial recognition (RF) has its foundations since 1883, thanks to its study there are techniques that are based on models and appearances. RF requires three stages: a) Face detection in an image, b) feature extraction, and c) identification and/or verification. This paper presents the methodology to carry out, through RF, the detection of the mask on the face. The first favorable results are presented, from which the bases for an optimization process that seeks to identify the correct use of the mask are proposed.

Keywords: COVID-19, facial recognition, mask detection.

1. Introducción

En el último año la sociedad ha sufrido un fuerte cambio en la forma de relacionarnos y de vivir nuestras vidas diarias. Esto ocasionado por la pandemia de COVID-19 donde la principal problemática ha sido la rápida propagación del virus. La situación nos obligó a aislarnos para evitar el contagio. Sin embargo, algunas personas como son los médicos corren un alto riesgo de contagio al estar expuestos al virus constantemente realizando sus trabajos, en muchos casos sin protección alguna. [1].

El reconocimiento facial es una tecnología que se encarga de identificar a un sujeto de forma no invasiva, con la ayuda de una cámara, esta tecnología se ha desarrollado con rapidez. Fue Alphonse Bertillon en 1883 quien sentó las bases del sistema de reconocimiento facial, usaba como base un sin número de medidas antropométricas como: la distancia entre los ojos, la simetría o los diferentes rasgos faciales de un individuo. Este sistema es una tendencia en el ámbito forense e incluso ha llegado al punto de tener opinión en la corte de justicia para determinar la inocencia de una persona con antecedentes penales. Su uso para reconocer personas puede agilizar el control de accesos a edificios corporativos y gubernamentales.

Esta tecnología se ha utilizado incluso para identificar criminales conocidos o sospechosos y en el ámbito socio económico, las empresas pueden analizar el rostro de los clientes para adaptar estrategias de marketing. Pero esta tecnología también enfrenta serios problemas de privacidad, ya que puede ser utilizada para rastrear a los individuos a través de sus comunidades, incluso por el mundo.

Los avances del reconocimiento facial parten del estudio de la biometría y ésta sumada a la tecnología, da como resultado la toma de medidas y el análisis de datos biológicos como el ADN, la huella dactilar, el iris y la voz. Así nacen los sistemas de reconocimiento facial, que toma sus decisiones de identificación con la ayuda de las características personales (fotografías y videos) de cada persona y plasmándolo en una imagen digital que puede ser reconocida o verificada de forma automatizada mediante un ordenador.

2. Estado del arte

En el proceso de reconocimiento facial se utilizan algoritmos para el procesamiento de imágenes donde se analizan cientos de rostros y se utiliza un mapeo facial que capta al rededor de100 expresiones faciales, todas tienen una medida de 50 x 50 píxeles de ancho y alto, por lo que el costo computacional aumenta [2].

Hay dos familias de técnicas de reconocimiento facial: técnicas basadas en la apariencia y técnicas basadas en modelos. Vea la figura 1.

2.1. Reconocimiento facial

Los sistemas basados en la apariencia se utilizan directamente sobre las imágenes, sin hacer uso de modelos 3D, cada imagen se representa como un punto en un subespacio vectorial para clasificar, para ello es necesario entrenar previamente el sistema que se utilizara con imágenes de diferentes rostros, con diferentes vistas. Por otro lado, los sistemas basados en modelos intentan construir un modelo lo más

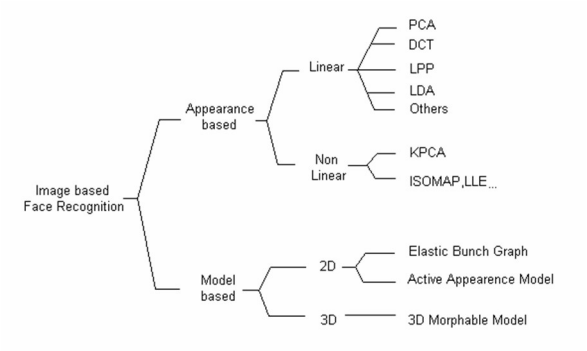

Fig. 1. Técnicas de reconocimiento facial. [3].

descriptivo posible de la cara humana, lo que permite detectar con precisión las variaciones faciales.

Estos sistemas tratan de obtener características biométricas de las imágenes para realizar el reconocimiento, el algoritmo sabe con anticipación el objeto que ha de representar y lo que intenta hacer es que corresponda el rostro real con el modelo. El proceso usado consiste en a) Construcción del modelo, b) Aplicación del modelo a la imagen de prueba y c) Uso de parámetros del modelo ajustado para calcular la similitud entre la imagen de prueba y las imágenes de referencia para realizar el reconocimiento.

Independientemente de la técnica utilizada para la solución, el RF requiere de tres etapas: a) Detección del rostro en una imagen, b) extracción de las características y c) identificación y/o verificación [5].

La detección del rostro implica encontrar las áreas dentro de una imagen que contiene un rostro. Básicamente se trata de descartar todo lo que sea fondo y así obtener la ubicación y tamaño exacto de la cara esto se puede lograr por medio de clasificación de patrones utilizando redes neuronales artificiales (perceptrón multicapa). La extracción de las características es la obtención de propiedades o parámetros particulares de cada rostro para luego poder ser clasificados.

El reconocimiento consiste en clasificar las características extraídas de cada rostro, para esto existen tres aproximaciones basadas en: a) concepto de similaridad, b) aproximación probabilista y c) optimización de criterio error.

Las aproximaciones basadas en el concepto de similaridad son las más simples e intuitivas, los patrones similares son asignados a la misma clase, se establece una métrica que define la similaridad y luego se clasifica por medio de plantillas o mínima distancia usando uno o varios prototipos de clase.

La técnica de eigenfaces aplica la regla del vecino más cercano, utilizando como métrica la distancia Euclídea. En el enfoque probabilístico, se utilizan los conceptos de la teoría de la decisión estadística para establecer los bordes de decisión de las diferentes clases. Se asume que las características, que representan a un patrón, tienen una función de densidad de probabilidad condicionada a la clase. Las reglas de decisión más conocidas: Bayes y la regla de máxima probabilidad.

La tercera aproximación se basa en construir los bordes de decisión optimizando algún criterio de error. Las redes neuronales son entrenadas con un algoritmo de entrenamiento a partir de un conjunto de datos (perceptrón multicapa). [3, 4].

2.2. Reportes de reconocimiento facial

En el Sistema de INTERPOL de Reconocimiento Facial (IFRS) se almacenan las imágenes faciales enviadas por más de 179 países, lo que la convierte en una base de datos policial de ámbito mundial única. Combinado con un software automatizado de identificación biométrica, este sistema es capaz de identificar a una persona o de comprobar su identidad mediante la comparación y el análisis de los modelos, formas y proporciones de sus rasgos y contornos faciales [6].

En la Universidad Técnica del Norte los estudiantes de la FICA (Facultad de Ingeniería en Ciencias Aplicadas) desarrollaron un prototipo que permite identificar sujetos conocidos (registrado en la base de datos) y no conocidos. Se usaron CNNs (Redes Neuronales convolucionales), y herramientas como un smartphone y una cámara de video vigilancia en tiempo real [7].

Un reporte sobre el diseño de un sistema de reconocimiento de rostros, aplicando inteligencia y visión artificial, usa un modelo para generar y establecer puntos de interés en el rostro, con el fin de entrenar una red neuronal, que permita identificar características faciales, para el reconocimiento y clasificación de sujetos. Se utilizó el algoritmo ASM (Active Shape Model – Modelo de Formas Activas) el cual genera un modelo de patrones y características que se pretenden determinar [8].

En el artículo Redes neuronales artificiales para el control de acceso basado en reconocimiento facial, se generó un prototipo de reconocimiento facial, utilizando un método basado en características geométricas. Primero obtuvieron los patrones del rostro de los usuarios para generar y construir las redes neuronales, cada determinado tiempo se obtienen variaciones de patrones del rostro de un usuario [9].

En [10] se generó un diseño de un sistema de detección de intrusos y control de acceso mediado por modelos biométricos e inteligencia artificial. Se utilizó como base el algoritmo de detección de rostros implementado por Geitgey, Deep Learning y visión computacional, apoyado sobre OpenCV. Convierte la imagen de entrada a escala de grises (0 - 255), se aplican los descriptores HOG, los cuales evalúan cada uno de los pixeles presentes en la imagen y determinan su gradiente con respecto a su entorno, para a partir de allí se realiza la cuantificación de los elementos.

En [3] se realizan pruebas con los métodos de reconocimiento facial, principalmente PCA, DCT y LPP usando bases de datos públicas para el análisis y desarrollo de las pruebas y obtención de resultados. El uso de estos métodos requiere un número de componentes para trabajar e identificar, para que estos métodos tengan el mejor rendimiento, en cada prueba se determinó que el número de componentes a utilizar según el método es: 55 para PCA, 80 para DCT y 40 para LPP, y se puede decir que el método más consistente y que ofrece un mayor índice de identificación correcta es LPP.

3. Metodología

Para llevar a cabo la detección del uso correcto del cubrebocas, se realizan las siguientes etapas en el sistema desarrollado: Detección de rostros en imágenes, clasificación de rostros, entrenamiento y reconocimiento facial. Una vez que se genera esta secuencia se puede aplicar a diferentes objetivos, como por ejemplo identificador de emociones, reconocimiento de accesorios, como lentes, cubrebocas, etc.

3.1. Detección de rostros

El programa para llevar a cabo el reconocimiento o detección de rostros se lleva a cabo con las librerías de OpenCV cv2 y os. Lo que realiza el programa es primero declarar la ruta donde se encuentran las imágenes en una variable llamada imagesPath y la creación de una carpeta llamada "Rostros encontrados".

Para el proceso de la detección de los rostros se utilizó el dataset proveniente dentro del mismo OpenCV conocido como "haarcascade_frontalface_default.xml".

Finamente inicializamos un contador y creamos un ciclo for que realizará la lectura de las imágenes con *imread(),* para convertirla a escala de grises. Esto nos permitirá comparar la imagen con el data set y determinar si coincide para cada imagen dentro del directorio de imagesPathList.

Posteriormente creamos la variable faces la cual recibirá la información de la función *faceClassif.detectMultiScale()* dentro de la cual tenemos gray que es la imagen convertida a escala de grises, el factor de escala y el número de detección que puede haber cerca una con otras.

Para los rostros que detecte el programa dentro de la variable faces se dibujara un rectángulo en su rostro respectivo *cv2. Rectangles ()* con la imagen de entrada, las medidas del rectángulo que en este caso serán las coordenadas x & y donde se encuentren los rostros, el color y el tipo de borde que tendrá que en este caso es -1 que significa que será toda la figura.

Finalmente, los rostros encontrados se guardarán con la función cv2.imwrite() en la ruta especificada dentro del parámetro que es ['Rostros encontrados/rostro_{}.jpg'. format(count), rostro] donde {} será el contador que lleve el proceso de rostros del código:

```
import cv2
import os
imagesPath = "C:/Users/vioro/Downloads/DELFIN_PROJECT/4. Codigos/
3_Deteccion de rostros/Directorio de muestras"
imagesPathList = os.listdir(imagesPath)
print('ImagesPathList= ',imagesPathList  )

if not os.path.exists('Rostros encontrados'):
    print('Carpeta creada: Rostros encontrados')
    os.makedirs('Rostros encontrados')
faceClassif =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface
_default.xml')
count = 0
for imageName in imagesPathList:
    #print('ImagenName= ',imageName)
    image = cv2.imread(imagesPath+'/'+imageName)
    #cv2.imshow('Imagen',image)
    imageAux = image.copy()
    gray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
    faces = faceClassif.detectMultiScale(gray, 1.1, 5)
    for (x,y,w,h) in faces:
        cv2.rectangle(image, (x,y),(x+w,y+h),(128,0,255),2)
    cv2.rectangle(image,(10,5),(450,25),(255,255,255),-1)
    cv2.rectangle(image,(10,5),(450,25),(255,255,255),-1)
    cv2.putText(image,'Presione R, para alamacenar los rostros
encontrados',(10,20), 2, 0.5,(128,0,255),1,cv2.LINE_AA)
    cv2.imshow('image',image)
    cv2.waitKey(0)
    k = cv2.waitKey(0)
    if  k == ord('R'):
```

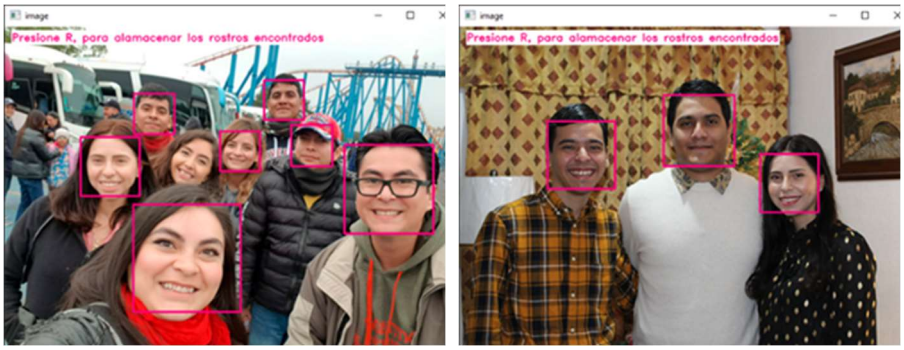

Fig.2. Programa para detección de rostros.

```
            for (x,y,w,h) in faces:
                rostro = imageAux[y:y+h,x:x+w]
                rostro = cv2.resize(rostro,(150,150), interpolati
            on=cv2.INTER_CUBIC)
                #cv2.imshow('rostro',rostro)
                #cv2.waitKey(0)
                cv2.imwrite('Rostros encontrados/rostro_{}.
            jpg'.format(count),rostro)
                count = count +1
        elif k == 27:
            break
    cv2.destroyAllWindows()
```

3.2. Clasificación de rostros

Para iniciar el reconocimiento facial por medio de visión artificial realizamos 3 programas; *Clasificador_Rostros.py* para crear nuestra data set que se utilizara y que proviene prácticamente del programa de detección de rostros anterior, el Entrenamiento.py para entrenar a la computadora y pueda identificar a la persona y por último *Reconocimiento_Facial.py* para la detección final.

El programa *Clasificador_Rostros.py* lo utilizaremos para que la cámara detecte a un usuario y lo almacene en una carpeta con su ID o identificador que en este caso es el nombre.

Utilizando además de las librerías cv2 y os utilizamos imutils. Primero, declaramos el nombre de la persona que queremos identificar en personalName. Ahora establecemos la ruta de la carpeta donde estarán todas las personas a identificar y una nueva variable llamada personPath que será la ruta establecida en dataPath y el nombre almacenado en personName.

Capturamos la información de la cámara dentro de la variable cap e inicializamos la variable faceClassif que utilizara el dataset para el reconocimiento de los rostros de haarcascades que antes se explicó.

Inicializamos el contador y con un ciclo **while** almacenamos la información de video dentro de la variable frame, la cual se modificará con la función de imutils para cambiar el tamaño del video y la conversión a escala de grises. Ahora utilizando el mismo código del programa anterior para detección de rostros:

```
import cv2
```

```
import os
import imutils
personName = 'Rodri'
dataPath = 'C:/Users/vioro/Downloads/DELFIN_PROJECT/4.
Codigos/Directorio de muestras II/Data'
personPath = dataPath + '/' + personName
if not os.path.exists(personPath):
    print('Carpeta creada: ',personPath)
    os.makedirs(personPath)
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
faceClassif =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface
_default.xml')
count = 0
while True:
    ret, frame = cap.read()
    if ret == False: break
    frame =  imutils.resize(frame, width=640)
    gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
    auxFrame = frame.copy()
    faces = faceClassif.detectMultiScale(gray,1.3,5)
    for (x,y,w,h) in faces:
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,255,0),2)
        rostro = auxFrame[y:y+h,x:x+w]
        rostro = cv2.resize(rostro,(150,150),interpolatio
       n=cv2.INTER_CUBIC)
        cv2.imwrite(personPath + '/rotro_{}.jpg'.format(c
       ount),rostro)
        count = count + 1
    cv2.imshow('frame',frame)
    k =  cv2.waitKey(1)
    if k == 27 or count >= 500:
        break
cap.release()
cv2.destroyAllWindows()
```

3.3. Entrenamiento de rostros

Ahora pasamos al programa de Entrenamiento.py que se encarga del entrenamiento de la visión artificial para la detección del usuario. En este programa tendremos 3 métodos distintos; Eigen Face, Fisher Face y LBPH Face.

Antes de continuar explicaremos un poco los métodos:

- El método de Eigen Faces emplea el método PCA (Principal Component Analysis), donde la idea principal es que un conjunto de datos de alta dimensión a menudo se describe mediante variables correlacionadas y, por lo tanto, solo unas pocas dimensiones significativas explican la mayor parte de la información.
- El método de Fisher Faces Este método es una mejora del anterior.

El método LBPH (Histogramas de Patrones Binarios Locales) presenta mejoras respecto a los métodos anteriores, ya que es más robusto ante cambios de iluminación. Además, dentro de la documentación de OpenCV la idea es no mirar la imagen completa como un vector de alta dimensión, sino describir solo las características locales de un objeto.

Es importante considerar que en cualquiera de los 3 métodos anteriores las imágenes de entrenamiento como de predicción deben estar en escala de grises, además de que en el método Eigenfaces se requiere que las imágenes de entrenamiento y prueba sean del mismo tamaño.

Enseguida se muestran los 3 programas mencionados:

```
ENTRENAMIENTO 1 (Eigen Face)
face_recognizer = cv2.face.EigenFaceRecognizer_create()
print("Procesando informacion...")
face_recognizer.train(facesData, np.array(labels))
face_recognizer.write('modeloEigenFace.xml')
print("Proceso finalizado...")
cv2.destroyAllWindows()

ENTRENAMIENTO 2 (Fisher Face)
face_recognizer = cv2.face.FisherFaceRecognizer_create()
print("Procesando informacion...")
face_recognizer.train(facesData, np.array(labels))
face_recognizer.write('modeloFisherFace.xml')
print("Proceso finalizado...")
cv2.destroyAllWindows()

ENTRENAMIENTO 3 (LBPH Face)
face_recognizer = cv2.face.LBPHFaceRecognizer_create()
print("Procesando informacion...")
face_recognizer.train(facesData, np.array(labels))
face_recognizer.write('modeloLBPHFace.xml')
print("Proceso finalizado...")
cv2.destroyAllWindows()
```

3.4. Reconocimiento facial

En este proceso ahora se explica el programa de reconocimiento, ya que tenemos el archivo entrenado que detectará a la persona podemos explicar el ultimo código (*Reconocimiento_Facial.py*).

Inicializamos la variable de face_recognizer con la función de *cv2.face.FisherFaceRecognizer_create()* y proseguimos a abrir el archivo entrenado xml con la función de *fase_recognizer.read()* con el nombre del archivo xml generado del programa anterior y empezamos a almacenar el video de la cámara en la variable cap, en seguida inicializamos la variable faceClassif que nos ayudara a identificar el rostro con el dataset de haarcascades. Continuamos con un ciclo while para la variable frame, con un cambio de escala. Iniciaclizamos la variable de la cara llamada faces con *faceClassif.detectMultiScale (gray, 1.3,5)* que detectara el rostro del video a escala de grises con un con sus parámetros establecidos para la detección.

Finalmente para cada faces obtenida mediante la función for la variable auxFrame pasara a la variable rostro con una reposicionamiento y un re-escalamiento a 150x150 pixeles con una interpolación simple cubica con la función antes explicada de *cv2.resize(),* con las imágenes generadas en el primer programa de detección y el video entrante utilizamos la función *fase_recognizer.preditc()* con la variable rostro dentro de su paréntesis para comparar las imágenes y buscar una coincidencia, si existe una o no se guardara en una variable llamad resultado en forma de un valor numérico. Aunque se pueden utilizar las 3 opciones de los métodos, utilizaremos el método de Fisher Face para comparar los resultados. Creamos una condición if para resultado comparado con 500:

```
face_recognizer = cv2.face.EigenFaceRecognizer_create()
face_recognizer.read('modeloEigenFace.xml')
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
faceClassif =
cv2.CascadeClassifier(cv2.data.haarcascades+'haarcascade_frontalface
_default.xml')
while True:
```

```
  ret,frame = cap.read()
  if ret == False:break
  gray = cv2.cvtColor(frame, cv2.COLOR_BGR2GRAY)
  auxFrame =gray.copy()
  faces = faceClassif.detectMultiScale(gray, 1.3,5)
  for (x,y,w,h) in faces:
    rostro = auxFrame[y:y+h,:x+w]
    rostro = cv2.resize(rostro,(150,150),interpolation =
cv2.INTER_CUBIC)
    result = face_recognizer.predict(rostro)
    cv2.putText
(frame,'{}'.format(result),(x,y+5),1,1.3,(0,255,120),1,cv2.LINE_AA)
    # EigenFaces
    if result[1] < 500:
        color 0 (0,255,120) if LABELS results [0]
    else:
        cv2.putText(frame,'Desconocido',(x,y-
20),2,0.8,(0,0,255),1,cv2.LINE_AA)
        cv2.rectangle(frame, (x,y),(x+w,y+h),(0,0,255),
       2)
  cv2.imshow('Video',frame)
  k = cv2.waitKey(1)
  if k== 27:
    break
cap.release()
cv2.destroyAllWindows()
```

3.5. Detección de mascarilla

Ahora para la variable de detección que es *mp_face_detection* utilizaremos una función de la librería media pipe que es *mp.solutions.face_detection*.

Definiremos un arreglo llamado labels los cuales tendrá los nombres de las carpeteas con las imágenes de dataset que son Con_Mascarilla y Sin_Mascarilla e inicializamos la variable que reconocerá este dataset que es face_mask con *cv2.face.LBPHFace Recognizer_create()* y *face_mask.read("face_mask_model.xml")* que es el método utilizado y el archivo generado, posteriormente dentro de la función *mp_face_detection.FaceDetection()* aplicamos un valor de confianza que será 0.5 para la variable face_detection.

Guardamos la información de la imagen del video de la cámara con frame.shape en dos variables que se llamaran height y width que almacenara el tamaño del video, inicializamos la variable llama results que obtendrá su resultado de la función *face_detection.process(frame_rgb)* que detectara si existe alguna coincidencia del video entrante convertido a RGB con la información entrenada en la carpeta data del dataset. Utilizando un condicional if para definir que si el *results.detections* tiene coincidencias obtendrá las coordenadas del rostro con la función *detection.location_data.relative_ bounding_box.[variable a pedir] *[ancho o alto (height o width)]*.

Finalmente se utilizara la función *face_mask.predict()* para la variable fase_image para detectar si existe alguna coincidencia positiva, este resultado lo guardara en la variable result. Los resultados solo podrán ser dos: “Con_mascarilla” o “Sin_mascarilla” que fue como se entrenó al programa.

```
mp_face_detection = mp.solutions.face_detection
#mp_drawing = mp.solutions.drawing_utils

LABELS = ["Con_Mascarilla", "Sin_Mascarilla"]
face_mask = cv2.face.LBPHFaceRecognizer_create()
face_mask.read("face_mask_model.xml")
```

```
cap = cv2.VideoCapture(0,cv2.CAP_DSHOW)
with mp_face_detection.FaceDetection(min_detection_confidence=0.5)
as face_detection:

    while True:
        ret,frame = cap.read()
        if ret == False: break
        frame= cv2.flip (frame,1)
        height, width, _ = frame. shape
        frame_rgb =  cv2.cvtColor(frame, cv2.COLOR_BGR2RGB)
        results = face_detection.process(frame_rgb)
    # ---------------------------------------    Dibujo de los
resultados
        if results.detections is not None:
            for detection in results.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)
#cv2.rectangle(frame,(xmin,ymin),(xmin + w ,ymin + h),(0,200,120),2)
                if xmin < 0 and ymin < 0:
                    continue


face_image = frame [ymin: ymin + h , xmin : xmin + w]
face_image = cv2.cvtColor(face_image, cv2.COLOR_BGR2GRAY)
face_image = cv2.resize(face_image,(72,72),
interpolation=cv2.INTER_CUBIC)
                #cv2.imshow('Zoom',face_image)


                result = face_mask.predict(face_image)
                #cv2.putText (frame,'{}'.format(result),(xmin ,ymin
+ 5),1,1.3,(250,250,250),1,cv2.LINE_AA)


                if result[1] < 150:
                   color = (0, 255,120) if LABELS[result[0]] ==
"Con_Mascarilla" else (0, 0, 250)
cv2.putText(frame,'{}'.format(LABELS[result[0]]),(xmin, ymin - 25),
2, 1, color,1,cv2.LINE_AA)
cv2.rectangle(frame, (xmin,ymin),(xmin+w, ymin+h),color,2)


        cv2.imshow('Video',frame)
        k = cv2.waitKey(1) & 0xFF
        if k == 27:
            break


cap.release()
cv2.destroyAllWindows()
```

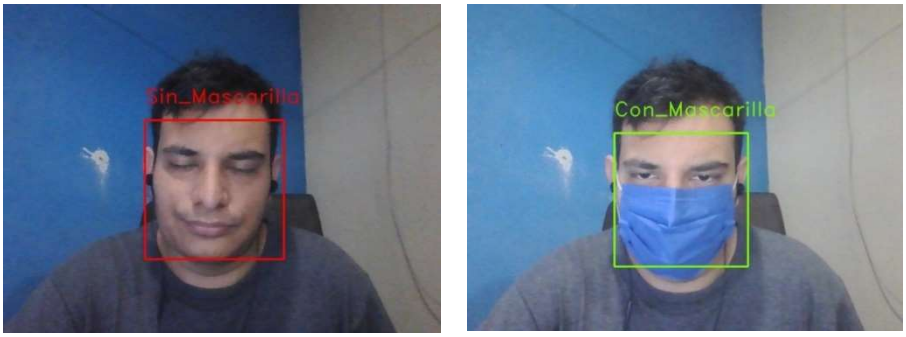

Fig. 3. Resultados del programa para la detección de la mascarilla.

En la figura 3 se muestran los resultados del programa para la detección de la mascarilla.

4. Resultados

En el caso del tiempo de entrenamiento para dos personas fue de aproximadamente 50 segundos aproximadamente con los primeros 2 métodos y de 25 segundos para el último método con un procesador AMD A-10 7860k (quad core a 4.0 GHz) y 16 RAM DDR3 1600MHz. Pero al incluir a una tercera persona los tiempos aumentaron a 5 minutos para los 2 dos métodos y a 1 minuto para el último método.

En cuanto a los resultados obtenidos por los 3 métodos utilizados el tercero de LBPH funciono de mejor manera siendo más estable demás de ser más rápido y fluido en comparación con los otros dos. Tanto el EigenFaces como Fisher Faces presentaban algunos falsos positivos mientas el otro no La precisión depende mucho de la iluminación que se tenga en la habitación y que refleja al usuario pen en general se obtuvieron resultados satisfactorios.

El resultado es muy satisfactorio y estable, además al haber utilizado un dataset ya hecho con muchas muestras con diferentes propiedades físicas podemos utilizar el programa con cualquier persona y de igual manera funcionara correctamente.

Referencias

1. Velavan, T. P., Meyer, C. G.: The COVID-19 epidemic. Tropical Medicine & International Health: TM & IH, vol. 25, no. 3, pp. 278–280 (2020)
2. Moreano, J. A. C., Pulloquinga, R. H. M., Lagla, G. A. F., Chisag, J. C. C., Pico, O. A. G.: Reconocimiento facial con base en imágenes. Boletâin Redipe, vol. 6, no. 5, pp. 143–151 (2017)
3. Gimeno-Hernández, R.: Estudio de técnicas de reconocimiento facial. Upc.Edu (2021) from https://upcommons.upc.edu/bitstream/handle/2099.1/9782/PFC_Roger Gimeno.pdf
4. Scarel, A., Martinez, G. M. D., Stegmayer, C. C. D. D., Muller, G. A. T.: Sistema de reconocimiento facial. Edu.ar (2021) http://sinc.unl.edu.ar/sincpublicatio ns/2010/SMS10/sinc_SMS10.pdf

5. Interpol: Reconocimiento facial (2021) https://www.interpol.int/es/Como-trabajamos/Policia-cientifica/Reconocimiento-facial
6. Chacua-Criollo, B. E.: Diseño de un sistema prototipo de reconocimiento facial para la identificación de personas en la facultad de ingeniería en ciencias aplicadas (FICA) de la Universidad Técnica del Norte utilizando técnicas de inteligencia artificial. Tesis de pregrado, Universidad Técnica del Norte. http://repositorio.utn.edu.ec/handle/123456789/9572 (2019)
7. Gualdrón, O. E., Suárez, O. M. D., Rojas, M. A. C.: Diseño de un sistema de reconocimiento de rostros mediante la hibridación de técnicas de reconocimiento de patrones, visión artificial e IA, enfocado a la seguridad e interacción robótica social. Mundo FESC, vol. 3, no. 6, pp. 16–28 (2013) http://www.revistapuce.edu.ec/index.php/revpuce/article/view/140/242
8. Porras, I. J. M.: Diseño de un sistema de reconocimiento facial como medio de control de acceso biométrico mediado por técnicas de inteligencia artificial como herramienta base de seguridad del CEAD IBAGUÉ. (2021) https://repository.unad.edu.co/bitstream/handle/10596/22930/Juan.Aldana.pdf?sequence=1&isAllowed=y