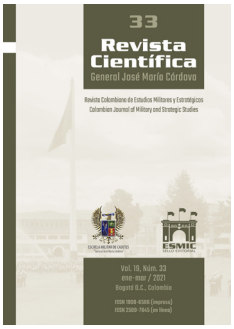

Revista Científica General José María Córdova
(Revista colombiana de estudios militares y estratégicos)
Bogotá D.C., Colombia

ISSN 1900-6586 (impreso), 2500-7645 (en línea)
Web oficial: https://www.revistacientificaesmic.com

Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección

Alexandre Pereira Junior
https://orcid.org/0000-0002-9438-5858
a.junior@aluno.ifsp.edu.br
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Brasil

Thiago Pedro Donadon Homem
https://orcid.org/0000-0003-2140-0129
thiagohomem@ifsp.edu.br
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Brasil

Fabio Oliveira Teixeira
https://orcid.org/0000-0002-1031-2644
fabio.teixeira@ifsp.edu.br
Instituto Federal de Educação Ciência e Tecnologia de São Paulo, Brasil

Citación: Junior, A. P., Homem, T. P. D., & Teixeira, F. O. (2021). Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección. *Revista Científica General José María Córdova*, *19*(33), 205-222. http://dx.doi.org/10.21830/19006586.725

Publicado en línea: 1.º de enero de 2021

Para enviar un artículo:
https://www.revistacientificaesmic.com/index.php/esmic/about/submissions

ESMIC
SELLO EDITORIAL

Miles Doctus

Revista Científica General José María Córdova
(Revista colombiana de estudios militares y estratégicos)
Bogotá D.C., Colombia

Volumen 19, número 33, enero-marzo 2021, pp. 205-222
http://dx.doi.org/10.21830/19006586.725

Aplicación de inteligencia artificial para monitorear el uso de mascarillas de protección

Artificial intelligence application to monitor the use of protective masks

Alexandre Pereira Junior, Thiago Pedro Donadon Homem y Fabio Oliveira Teixeira
Instituto Federal de Educação, Ciência e Tecnologia de São Paulo, Brasil

Resumen. En el contexto de la pandemia actual, esta investigación crea una aplicación web que permite monitorear el uso de la mascarilla protectora en ambientes públicos. Utilizando el *framework* Flask, en el lenguaje de Python, la aplicación cuenta con un panel de control que ayuda a visualizar los datos obtenidos. El proceso de detección utiliza el algoritmo Haar Cascade para clasificar rostros con y sin mascarillas protectoras. Como resultado, la aplicación web es liviana y permite detectar y almacenar en la nube las imágenes capturadas y la posibilidad de un mayor análisis de datos. El clasificador presenta precisión, revocación y *f-score* de 63 %, 93 % y 75 %, respectivamente. Aunque la precisión fue satisfactoria, se realizarán nuevos experimentos para explorar nuevas técnicas de visión por computadora, como el uso del aprendizaje profundo.

Palabras clave: aplicación informática; COVID-19; detección facial; Haar Cascade; inteligencia artificial; prevención

Abstract. This work presents a web application that monitors face mask use in public environments in the current pandemic context. The application was executed using the Flask *framework* and the Python language. It has a control panel that helps the user visualize the data obtained. The monitoring process uses the Haar Cascade algorithm to classify faces with and without face masks. The web application is lightweight. It allows the detection and storage of the captured images on the "cloud" and the possibility of further data analysis. The classifier has a precision, revocation, and *f-score* of 63%, 93%, and 75%, respectively. Although the precision was satisfactory, further experiments will be conducted to explore new computer vision techniques, such as deep learning.

Keywords: artificial intelligence; COVID-19; facial detection; Haar Cascade; prevention; software application

Sección: Dosier • Artículo de investigación científica y tecnológica

Recibido: 5 de octubre de 2020 • Aceptado:3 de diciembre de 2020

Contacto: Thiago Pedro Donadon Homem ✉ thiagohomem@ifsp.edu.br

Introducción

A finales del año 2019, en la ciudad de Wuhan se registraron los primeros casos de una enfermedad aún no conocida en el mundo. Las personas afectadas presentaban tos, estornudos, fiebre y, en el peor de los casos, una neumonía grave que provocaba insuficiencia respiratoria aguda (Vieira et al., 2020). La condición clínica de los pacientes evolucionó rápidamente de manera negativa y fueron necesarias las unidades de cuidados intensivos para intentar recuperar su estado de salud. En pocos días, Wuhan se convirtió en el primer epicentro de este nuevo virus y el mundo siguió sus esfuerzos para contener el brote, como, por ejemplo, con la construcción de hospitales en tiempo récord y medidas de distancia social severamente implementadas. Un ejemplo de estas medidas ha sido la cuarentena, cuyo objetivo es confinar a las personas para que no entren contacto con individuos infectados, lo cual es necesario en el caso de una enfermedad infecciosa que no tiene vacuna ni cura. Así, se aisló la ciudad de Wuhan para contener la proliferación del virus.

El nuevo virus recibió el nombre de SARS-CoV-2. Fue identificado como parte de la familia de los coronavirus, causante de la nueva enfermedad llamada COVID-19, cuya propagación por el mundo ha provocado un estado de emergencia de salud pública de importancia internacional, caracterizada por la OMS como una pandemia.

Las medidas de aislamiento fueron necesarias porque la tasa de transmisión del virus es muy alta; ocurre de una persona a otra a través del tacto, gotículas de saliva, estornudos, tos, flemas, objetos o superficies contaminadas. Según la Organización Mundial de la Salud (OMS), los cuidados básicos de higiene personal pueden reducir la tasa de trasmisión del virus, como la higiene de manos con alcohol y gel, además del uso de equipos de protección personal, como mascarillas quirúrgicas (N95) o mascarillas hechas a mano. Los estudios demuestran que las mascarillas tienen la capacidad de bloquear entre un 95,15 % y un 99,98 % de aerosoles (Reis, 2020).

Por otra parte, la tasa de mortalidad del COVID-19 es aproximadamente del 3,4 %, una tasa menor en comparación con otros coronavirus, como el SARS y el MERS, que tienen un porcentaje de casos fatales del 9,6 % y 34,3 %, respectivamente (Hewings-Martin, 2020). En algunos casos, la evolución clínica del virus requiere el uso de unidades de cuidados intensivos (UCI) con soporte ventilatorio. Debido a la velocidad de transmisión de la enfermedad, algunos países enfrentan dificultades para satisfacer la demanda de estos equipos y evitar el mayor número de muertes posible. Por tanto, las medidas de aislamiento social y protección individual han sido indispensables para reducir el número de infectados y no sobrecargar los sistemas de salud. Los países que pospusieron tales medidas, como Italia y España, enfrentaron hacinamientos en los hospitales, lo que condujo a miles de casos y de muertes por falta de camas para la atención, y en últimas llevó a la difícil situación de tener que decidir quién recibiría la atención adecuada y quién no.

En Brasil, las primeras acciones tomadas para frenar la propagación de la enfermedad se implementaron en los aeropuertos, evitando el ingreso de personas que presentaban los

síntomas característicos de la enfermedad, como tos, fiebre, secreción nasal, dolor de garganta, dificultad respiratoria, pérdida del olfato (anosmia), alteración del gusto (ageustia), trastornos gastrointestinales (náuseas, vómitos, diarrea), cansancio (astenia), disminución del apetito (hiperoxia) y disnea (dificultad para respirar). Sin embargo, la entrada del virus fue inevitable, porque en una determinada etapa de la enfermedad el individuo infectado es asintomático.

El estado de São Paulo se convirtió en el epicentro de la enfermedad en el país, debido a la densidad de población y el tráfico de personas. La ciudad tiene los aeropuertos más transitados de América del Sur, que sirven como puente aéreo para varios vuelos. Luego de la confirmación de los primeros casos de COVID-19, el estado declaró la cuarentena. Se impidió que las instituciones educativas y las actividades no esenciales mantuvieran su funcionamiento, sin previsión de reapertura.

La ciudad más grande del estado de São Paulo, mediante el Decreto 59396 (Diário Oficial de São Paulo, 2020), obligó a todas las personas a usar mascarillas en lugares públicos, comercios y transporte público, con una multa por incumplimiento. Las mascarillas se han convertido en un elemento esencial para los paulistas que necesitan moverse por la ciudad durante la emergencia. Porque incluso con los esfuerzos del gobierno de São Paulo, la tasa de aislamiento de la población es baja; en el periodo comprendido entre abril y mayo de 2020, osciló entre 46 % y 59 % (SIMI-SP, 2020), cuando lo ideal es que fuera del 70 %.

Los datos para medir la adherencia a la cuarentena se están recolectando a través de un sistema que mapea el índice de desplazamiento de la población, llamado Sistema de Monitoreo Inteligente de São Paulo (SIMI-SP, 2020). Este sistema fue construido a través de una iniciativa público-privada e involucró a los operadores móviles del estado. Si bien SIMI-SP es eficaz para identificar aglomeraciones y medir el porcentaje de aislamiento, no monitorea el comportamiento de la población durante su tránsito en lugares públicos, como, por ejemplo, en cuanto al uso de mascarillas protectoras.

Para complementar las acciones del SIMI-SP, es necesario desarrollar un sistema de monitoreo capaz de identificar si un individuo utiliza equipos de protección personal en lugares públicos donde circula gente. Las acciones de esta naturaleza serán necesarias mientras no se encuentre un fármaco o vacuna eficaz contra el COVID-19.

El comportamiento de las personas en tránsito durante la pandemia se puede monitorear mediante el uso de inteligencia artificial (IA). Según Pontes (2011), la IA es el área de la computación que busca simular las capacidades para resolver problemas y tomar decisiones. Su subárea, *computer vision*, es, según Backes y Junior (2019), "el área de estudio que intenta trasmitir a las máquinas la increíble capacidad de visión", ya que la computadora interpreta los datos de manera diferente a un ser humano. La visión artificial se centra en

> […] capturar imágenes y mejorarlas (por ejemplo, eliminar el ruido, aumentar el contraste, etc.), separar regiones u objetos de interés de una escena, extraer información

diversa en función de la imagen analizada como, por ejemplo, forma, color, textura y finalmente, relacionar las imágenes con otras vistas previamente.

Un tipo de monitoreo mejorado por la IA es el reconocimiento facial. Se utiliza en varias áreas, desde las más estéticas —por ejemplo, para mejorar las fotos utilizando el rostro como elemento principal en aplicaciones como Instagram— hasta en otras áreas como la agricultura y la pericia policial (Almanza, 2018). El reconocimiento facial se puede utilizar para monitorear a una gran cantidad de personas al mismo tiempo, resaltando sus rostros y utilizando algoritmos para transformar estas imágenes en datos comprensibles para un sistema informático.

Presentación del problema

El alto nivel de contagio del SARS-CoV-2 ha hecho obligatorio el uso de mascarillas protectoras en varios países, como una alternativa efectiva y de bajo costo para contener la propagación del virus, ya que protege tanto al individuo que lo usa como a quienes lo rodean. Según estudios, incluso con la disminución de casos, el uso de mascarillas y otras medidas como la distancia social pueden extenderse hasta 2022 (Kissler et al., 2020). Por lo tanto, son necesarias soluciones de monitoreo social autónomo que faciliten la reanudación de las actividades económicas y sociales asegurando el mantenimiento de las medidas durante y después del periodo más crucial de la pandemia.

Así, el objetivo principal de esta investigación es desarrollar una solución computacional utilizando técnicas de IA y visión por computadora para monitorear el uso de mascarillas protectoras en ambientes públicos y privados a partir de imágenes captadas por cámaras. Con ese fin, se proponen los siguientes objetivos específicos: 1) construir un sistema de detección facial con visión artificial y algoritmos de IA; 2) desarrollar un banco de datos para almacenar los datos obtenidos, y 3) crear un tablero para monitorear los datos del sistema.

Estudio de viabilidad

Esta sección presenta iniciativas y estudios relacionados con el proyecto propuesto. Se presentan soluciones computacionales desarrolladas para minimizar el impacto de la pandemia de COVID-19.

Trabajos relacionados

Incluso sin utilizar tecnologías de detección facial, vale la pena monitorear SIMI-SP (Portal do Governo, 2020), un sistema de monitoreo utilizado actualmente por el gobierno de São Paulo. El sistema crea un mapa de aglomeraciones y permite el seguimiento de la tasa de aislamiento social en los centros urbanos y los lugares más propicios para la proliferación de nuevos casos. Fue creado a partir de una asociación público-privada

que involucra a operadores de telefonía móvil. No obstante, esta solución no es capaz de identificar si una persona está usando equipo de protección personal, como las mascarillas protectoras recomendadas por la OMS y de uso obligatorio en algunas ciudades.

Una búsqueda en septiembre de 2020 en la biblioteca virtual de artículos científicos de Web of Science mediante el descriptor "*face recognition*" arrojó 10 325 ocurrencias en los últimos cinco años. De este resultado, 5737 artículos fueron publicados en revistas científicas, 4371 en congresos científicos y 217 en otras modalidades. Solo 116 artículos, aproximadamente el 1,1 %, fueron creados por autores brasileños. Aunque representan una pequeña parte de los estudios, algunas instituciones brasileñas abordan el uso del reconocimiento facial en sus proyectos.

Una de estas publicaciones aborda la categorización de género de las personas en la imagen, identificando a través de sus rasgos físicos si alguien es hombre o mujer, todo en tiempo real y en un entorno no controlado, con factores que pueden perjudicar la precisión de la imagen (por ejemplo, la luminosidad del lugar). Para solucionar este problema, los autores simularon la comprensión humana del reconocimiento de géneros, creando una red neuronal que identifica estos patrones (Araujo & Rosito, 2018).

Otra publicación de científicos brasileños es el estudio sobre Deepfake (Botelho et al., 2018). Esta tecnología utiliza IA para crear videos falsos que pueden ocasionar falsos positivos en los sistemas de reconocimiento facial. Mediante el uso de redes neuronales convolucionales (CNN, del inglés *convolutional neural networks*), los sistemas de seguridad pueden beneficiarse de la solución presentada en el artículo.

Entre los artículos seleccionados, un estudio brasileño también compara métodos convencionales de aprendizaje automático y aprendizaje profundo (*deep learning*) (Finizola et al., 2019). Según la publicación, hubo un mejor desempeño en el aprendizaje en tareas de reconocimiento facial para modelos creados a partir de bases de datos de entrenamiento con pocas personas.

Justificación

De acuerdo con lo anterior, ante el escenario de una pandemia con alto riesgo de trasmisión por vías respiratorias, es necesario fiscalizar el uso de mascarillas por parte de la población en ambientes públicos, a fin de garantizar la efectividad de las medidas de prevención del contagio de COVID-19. Este articulo presenta las tecnologías utilizadas con el propósito de construir una solución para el monitoreo autónomo del uso de mascarillas de protección individual por parte de la población en entornos no controlados.

Arquitectura de soluciones

A continuación se describe la arquitectura de la solución propuesta en este proyecto, así como la relación entre componentes computacionales utilizados.

Algoritmo clasificador

El clasificador utilizado en la aplicación está basado en las técnicas propuestas por Viola y Jones (2001) y en el método *Haar Cascade* (Mordvintsev, 2013), hasta ahora especializado en el rostro frontal, de personas y otros. Sin embargo, el proceso de clasificación requiere procesar previamente las imágenes.

Las imágenes en el sistema informático son matrices con tres dimensiones, organizadas por colores, ancho y alto. En el paso de procesamiento de imágenes, se definen su altura y ancho predeterminados, de modo que todas tengan el mismo tamaño en pixeles. Durante el procesamiento de las imágenes, todavía es posible eliminar cierto ruido de la imagen obtenida en el proceso de captura, como los filtros de suavizado y resaltado. En este trabajo se decidió omitir la dimensión del color, convirtiendo la imagen en una escala de grises. De esta forma, el clasificador realizará funciones en una matriz con dos dimensiones.

El método Haar Cascade (Viola & Jones, 2001) consta de tres etapas fundamentales: 1) rastrea la imagen en busca de características Haar que se asemejen a los objetos buscados; 2) utiliza un clasificador *boosting* para seleccionar las características más relevantes, y 3) conecta en cascada los clasificadores, con el objetivo de mejorar los resultados finales.

Así, en una primera etapa, se escanea la imagen a través de pequeñas circunvoluciones de granos, de izquierda a derecha, de arriba a abajo, en busca de las características de Haar (Lienhart & Maydt, 2002). La Figura 1 ejemplifica los tipos de núcleos manuales utilizados para la detección de Haar Cascade: la imagen (a) hace referencia a los granos para identificar las características del borde; la imagen (b) hace referencia a los granos para identificar las características de línea, y la imagen (c) hace referencia a los granos como cuatro rectángulos, usados para reconocer bordes diagonales.

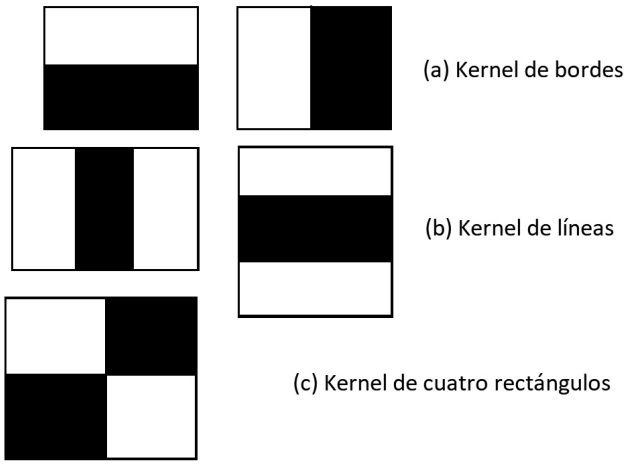

Figura 1. Núcleos utilizados en Haar Cascade.
Fuente: Adaptado de Mordvintsev (2013).

Los núcleos son matrices numéricas que representan las partes en blanco y negro de la Figura 1, de modo que, durante el escaneo, cada pixel de la imagen se multiplica por el número de núcleos correspondiente. La diferencia entre la suma de los pixeles de la parte blanca y la suma de los pixeles de la parte negra genera el valor de una característica. Al final de la proyección, se obtiene un conjunto de características.

La segunda etapa se basa en AdaBoost y tiene como objetivo seleccionar una pequeña cantidad de características visuales con mayor intensidad para una imagen determinada. Estas características visuales se utilizan para la detección y clasificación de objetos, lo cual se logra con base en algoritmos de impulso. El impulso es una técnica de aprendizaje automático que combina varios clasificadores débiles con el objetivo de mejorar la precisión general (Duarte, 2009).

La primera característica seleccionada está destinada a centrarse en la región de los ojos, ya que esta suele ser más oscura que la región de la nariz y las mejillas. La segunda característica seleccionada se basa en la propiedad de que los ojos son más oscuros que la punta de la nariz. Para cada característica visual, el método encuentra el parámetro para clasificar caras futuras positivas y negativas seleccionando los recursos con la menor tasa de error.

El proceso en cascada (del inglés *Cascade*) se lleva a cabo en la tercera etapa, mediante la combinación de clasificadores en forma de árbol degenerado. Esta estrategia tiene como objetivo reducir la tasa de falsos positivos mediante la combinación de un clasificador en cada etapa de la cascada. Para ejemplificar este proceso, se aplica un clasificador a partir de una imagen base, que descarta imágenes clasificadas como negativas, es decir, sin las características del objetivo. El siguiente paso considera solo las imágenes correctamente clasificadas y aplica otro clasificador rechazando nuevamente las imágenes negativas. El proceso en cascada continúa hasta una tasa de aciertos deseables. La Figura 2, adaptada de Harmouch (2020), ilustra el proceso de clasificación en cascada.

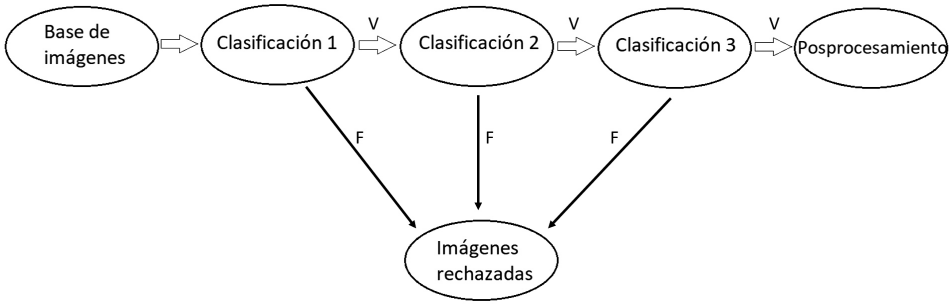

Figura 2. Proceso de clasificación en cascada. En cada etapa, una serie de imágenes clasificadas incorrectamente (F) son rechazadas y las clasificadas correctamente (V) siguen a la siguiente clasificación.
Fuente: Adaptado de Harmouch (2020).

De forma similar a lo que ocurre en los diversos métodos de aprendizaje supervisado, Haar también requiere la fase de formación. Para ello se debe crear una base de imágenes separadas por positivos (que contienen imágenes del objeto por identificar) y negativos (que no las contienen). Al finalizar el proceso de entrenamiento, el programa almacenará las características Haar del objeto en un archivo XML, que servirá de modelo para la detección de los objetos.

Para realizar el entrenamiento del clasificador, cada imagen recibe un peso igual al principio. Después de cada clasificación, se aumentan los pesos de las imágenes clasificadas incorrectamente. Luego, se realiza el mismo proceso calculando las tasas de error, así como nuevos pesos. Este proceso se repite hasta que se alcanza la precisión requerida según la tasa de error o hasta que se encuentra el numero requerido de recursos.

Haar Cascade es gratuito y está ubicado en un repositorio OpenCV, con clasificadores previamente entrenados en archivos XML. Así, su integración con el sistema es simple y flexible, lo que hace popular su implementación en proyectos de visión artificial para la detección de objetos.

Diagrama de componentes

La Figura 3 muestra la relación entre la aplicación y OpenCV Framework (OpenCV Team, 2020), el cual se encarga del tratamiento y codificación de imágenes. Se utilizó el clasificador Haar Cascade para detectar patrones en imágenes. Entre los beneficios que ofrece Haar Cascade, se destaca su fácil implementación y bajo consumo de recursos computacionales para entrenar y aprobar el modelo.

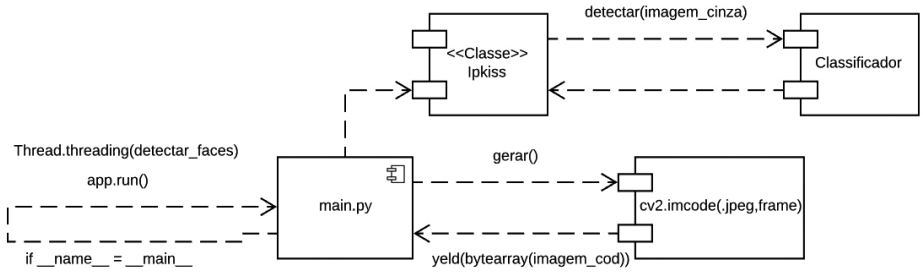

Figura 3. Componentes responsables de capturar y transformar imágenes.
Fuente: Elaboración propia.

La Figura 4 ilustra la conexión de la aplicación a la base de datos Atlas de MongoDB (MongoDB Inc., 2020). Esta es una base de datos no relacional, también conocida por la sigla NoSQL. Este tipo de base de datos mejora la velocidad de las transacciones y no prioriza propiedades relacionadas con atomicidad, consistencia, aislamiento y durabilidad (ACID), presentes en bases de datos relacionales. MongoDB Atlas es una base de datos orientada a documentos que comparte características similares con otros tipos de bancos,

como valores-clase, orientados a gráficos y columnas. Asimismo, utiliza recursos computacionales en la nube (*cloud computing*). Es un servicio caracterizado como DaaS (*Database as a service*) y permite que cualquier ordenador conectado a internet acceda a su sistema de gestión de base de datos y realice operaciones como inserción, borrado, alteración y consulta de registros. El modelo de implementación permite la adaptación de los recursos computacionales utilizados a las necesidades de la solución, como la velocidad de las transacciones y la flexibilidad de los esquemas que albergan los datos almacenados.

Por su parte, la biblioteca PyMongo (Dirolf, 2020) facilita la interacción entre el lenguaje de programación Python y la base de datos MongoDB. Contiene un conjunto de herramientas que permiten establecer las conexiones entre las capas de visualización y de datos de una aplicación web. Este proyecto utilizó la biblioteca PyMongo para construir los métodos de conexión, consulta, cambio, inserción y eliminación de registros en la base de datos.

Las imágenes captadas por la cámara fueron transformadas y, posteriormente, insertadas en la base de datos MongoDB Pillow (Clark et al., 2020). Luego IO (Python Software Foundation, 2020b) y Base64 (Python Software Foundation, 2020a) fueron los responsables de la tarea de transformación.

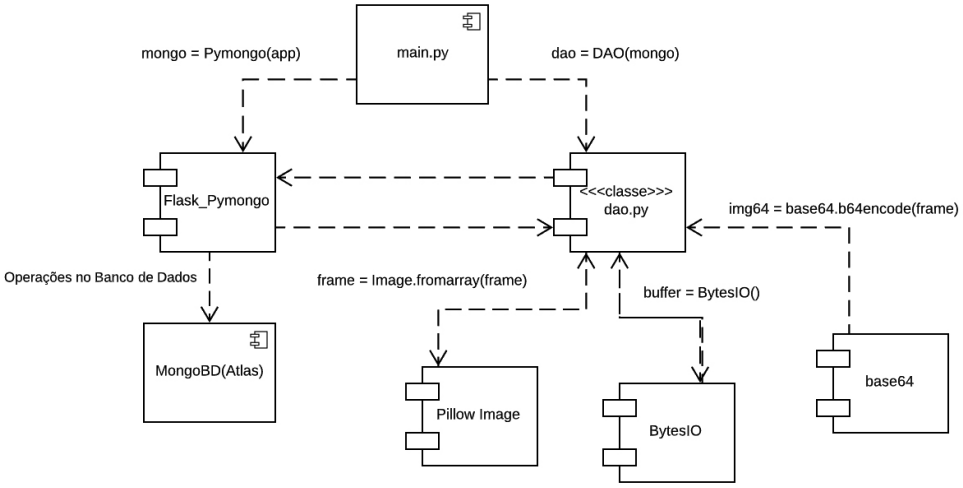

Figura 4. Componentes de las transacciones de la base de datos.
Fuente: Elaboración propia.

La Figura 5 muestra la relación entre los componentes de la capa *front-end* de la aplicación desarrollada en este proyecto. Se utilizó el micro-*framework* Flask (Pallets, s. f.) para orquestar las solicitudes procesadas por la aplicación. Este *framework* se eligió porque tiene una estructura optimizada y unos recursos adecuados a las necesidades del proyecto, además de que se adhiere al lenguaje Python. Otra ventaja es que tiene una modularidad

y flexibilidad que sirve a diferentes tipos de aplicaciones web, desde las más simples hasta las más complejas.

La usabilidad de la aplicación se ha mejorado mediante el uso de bibliotecas Javascript. Así, la biblioteca jQuery (OpenJS Foundation, 2020) se utilizó para crear solicitudes asincrónicas, evitando que la página se cargue varias veces durante la interacción del usuario. La biblioteca Chartist.js (Kunz, 2019), por otro lado, contribuyó a la construcción de gráficos receptivos y dinámicos, mientras que la biblioteca Materialize (Chang et al., 2020), creada por un grupo de estudiantes de la Universidad Carnegie Mellon con base en los principios de Material Design de Google, fue responsable de las hojas de estilo de la aplicación.

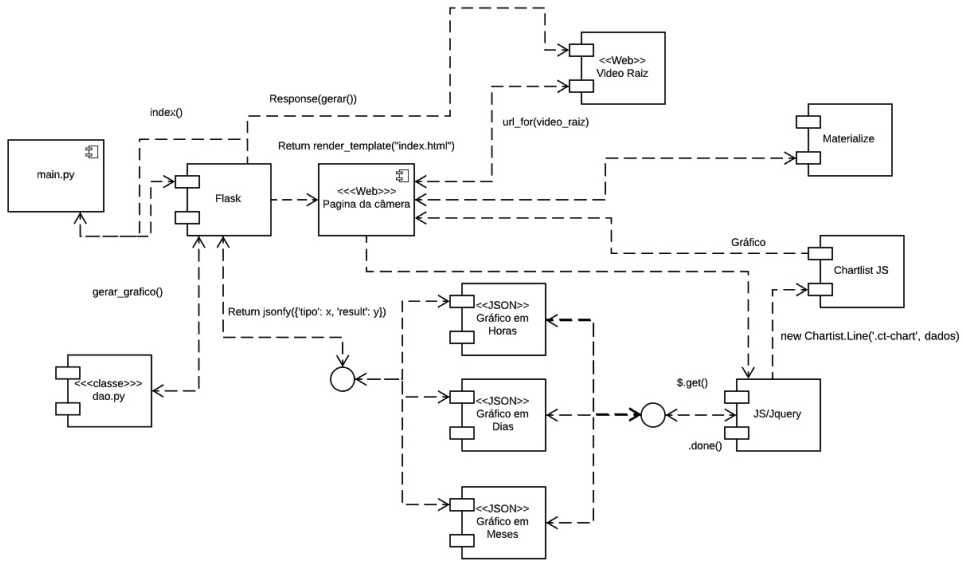

Figura 5. Componentes responsables de la visualización web.
Fuente: Elaboración propia.

El algoritmo 1 (Tabla 1) presenta la función encargada de capturar, detectar y almacenar el rostro. El algoritmo se ejecuta hasta que el usuario sale del programa (líneas 2 a 14); se captura una imagen de la cámara (línea 3); se redimensiona para acelerar el proceso de clasificación (línea 4); se convierte a escala de grises (línea 5), creando regiones oscuras y claras que facilitan la identificación de bordes o cambios en texturas; almacena la fecha y hora del sistema para el registro de base de datos (línea 6); detecta el rostro utilizando las características Haar (línea 7). Si se identifican una o más caras (línea 8), para cada cara detectada en la imagen (línea 9 a 12) realiza un corte de la cara en la imagen original (línea 10) y la cara cortada se almacena en una base de datos (línea 11).

Tabla 1. Algoritmo 1, que describe el proceso de detección y almacenamiento de rostros que no utilizan mascarillas

Algoritmo 1. Función Detecta y Almacena

```
1: Función Detecta
2:   Mientras (usuario no termina el programa): hacer
3:     capturar una nueva imagen img
4:     redimensionar img
5:     convertir img para la escala de grises imgg
6:     almacenar la fecha y hora local datetime
7:     identificar los rostros (faces) en imgg usando clasificador Haar Cascade
8:     Si el número de faces > 0 entonces
9:       Para cada faces ∈ img hacer
10:        cortar el rostro de img y almacenar en face
11:        almacenar en base de datos face y datetime
12:      Final Para
13:    Final Si
14:  Final Mientras
15: Final
```

Fuente: Elaboración propia.

Tecnologías utilizadas

La Tabla 2 muestra el resumen de las tecnologías utilizadas en este proyecto. La primera columna identifica la tecnología utilizada, la segunda columna corresponde a las capas de la aplicación web donde se necesita la tecnología. Estas capas se subdividieron en cliente, cuando la ejecución de funcionalidades ocurrió en la máquina del usuario, y servidor, cuando ocurrió en el servidor responsable de recibir solicitudes. La última columna ofrece la justificación para usar cada tecnología.

Tabla 2. Resumen de tecnologías utilizadas

Tecnología	Capa	Justificación
Git	Servidor	Control de versiones de código, guardando implementaciones en repositorios. Se requiere para el almacenamiento del proyecto.
Python	Servidor	Se necesita para la integración con bibliotecas de visión e inteligencia artificiales.

Continúa tabla...

Tecnología	Capa	Justificación
OpenCV	Servidor	Biblioteca principal para *computer vision*; entre todas las funciones que realiza, es la encargada de comunicarse con el clasificador.
Haar Cascade	Servidor	Clasificador encargado de detectar la presencia de un rostro en un fotograma de video.
Pillow	Servidor	Se requiere para escribir datos en el almacenamiento temporal.
BytesIO	Servidor	Se requiere para crear almacenamiento temporal.
Base64	Servidor	Se necesita para transformar un conjunto de bytes en texto comprensible en cualquier servidor o aplicación web.
ImUtils	Servidor	Se necesita para capturar la imagen de la cámara y cambiar el tamaño del marco.
Flask	Servidor	Micro-*framework* para desarrollo web.
HTML	Cliente	Patrón de páginas web.
Javascript	Cliente	Se requiere para realizar solicitudes desde la aplicación web al servidor y la base de datos.
jQuery	Cliente	Se requiere para simplificar las funciones Javascript.
Chartist JS	Cliente	Se necesita para crear gráficos simples y receptivos.
Materialize	Cliente	Es responsable del diseño generado por las hojas de estilo.
PyMongo	Servidor	Se requiere para la interacción de la aplicación con MongoDB.
MongoDB Atlas	Servidor	DBMS no relacional y orientado a documentos sin cargo.

Fuente: Elaboración propia.

Resultados y discusión

Se utilizó un modelo previamente entrenado de algoritmo Haar Cascade que está disponible en la biblioteca OpenCV. El paso de validación del algoritmo implementado en este proyecto recurrió a una base de datos con 194 imágenes (disponibles en DataFlair, 2020). Las imágenes utilizadas se distribuyeron equitativamente en dos clases: una clase tenía imágenes del rostro de personas que usaban mascarillas de protección individual y la otra tenía imágenes de rostros de personas sin mascarilla. La ejecución del algoritmo Haar Cascade para detectar rostros que usaban o no mascarillas de protección se realizó mediante medidas de precisión, revocación y f_β-*score* (Gehanno et al., 2009).

La precisión, revocación y f_β-*score* se basaron en los valores obtenidos por el clasificador a partir de las ocurrencias en medidas de verdaderos positivos (TP, del inglés *true*

positive), falsos positivos (FP, *false positive*), verdaderos negativos (TN, *true negative*) y falsos negativos (FN, *false negative*). Las ocurrencias atribuidas a la categoría TP se refieren al número de clasificaciones correctas de imágenes con caras sin mascarilla; el valor de FP se refiere a ocurrencias de clasificación incorrecta de imágenes con rostros sin mascarilla; el valor del TN se refiere a ocurrencias de la clasificación correcta de imágenes con rostros y mascarillas, y finalmente el valor FN se refiere a la ocurrencia de una clasificación incorrecta de imágenes con rostros y mascarillas. La Tabla 3 muestra la matriz de confusión y distribución de dichos valores.

Tabla 3. Matriz de confusión

		Valor verdadero	
		Clase positiva	**Clase negativa**
Predicción	Clase positiva	90 (TP)	54 (FP)
	Clase negativa	7 (FN)	43 (TN)

Fuente: Elaboración propia.

A partir de estos valores, la precisión se define mediante el cálculo de la ecuación 1, donde TP se divide por la suma de TP y FP:

$$P = \frac{TP}{TP + FP} \qquad \textit{Ecuación 1}$$

Por su parte, la revocación se define dividiendo las evaluaciones de TP por la suma de TP y FN, como se muestra en la ecuación 2:

$$R = \frac{TP}{TP + FN} \qquad \textit{Ecuación 2}$$

A su vez, el f_β*-score* (Magdy & Jones, 2010), representado en la ecuación 3, es la media armónica entre precisión y revocación, cuyos valores representan respectivamente los parámetros P y R. El parámetro β se puede utilizar para asignar diferentes pesos a las medidas utilizadas en la ecuación. En la evaluación de desempeño utilizada en este proyecto, β recibió un valor igual a 1; por lo tanto, la precisión y la renovación tienen la misma importancia.

$$f_\beta score = \frac{(1 + \beta^2) \cdot (P \cdot R)}{(\beta^2 \cdot P + R)} \qquad \textit{Ecuación 3}$$

Así, a través de las ecuaciones 1, 2 y 3 fue posible obtener los valores de precisión, revocación y f_1*-score* representados en la Tabla 4.

Tabla 4. Resultados obtenidos de precisión, revocación y f_1-*score*

Precisión	Revocación	f_1-score
0,63	0,93	0,75

Fuente: Elaboración propia.

Además de la evaluación del algoritmo Haar Cascade, se construyó un producto mínimo viable (MVP, por sus siglas en inglés) utilizando las tecnologías que se muestran en la Tabla 1. Este prototipo funcional de una aplicación web demostró el funcionamiento de las tecnologías y el *hardware* evaluados en este proyecto de forma acoplada y sincronizada.

La Figura 6 muestra la pantalla inicial del prototipo desarrollado. La región demarcada con línea discontinua en azul muestra el video en tiempo real, capturado por la cámara y enviado al algoritmo de visión por computadora. La segunda región, punteada en verde claro, muestra un carrusel de datos con rostros de los individuos, extraído del video, que no portaban mascarillas protectoras. La tercera región, punteada en naranja, muestra el gráfico cuantitativo de la ocurrencia de individuos que no usaron mascarillas en un periodo determinado (mensual, semanal o diario)[1].

Figura 6. Prototipo del sistema web desarrollado.
Fuente: Elaboración propia.

1 Un video de demostración del MVP está disponible en este enlace: https://youtu.be/QmCs_piHZkw. Se puede acceder al repositorio de soluciones en https://github.com/Eskandar1/Ipkiss

Discusión

Este apartado discute los resultados obtenidos en las pruebas, ya que, si estas se replican con otras técnicas, los resultados pueden ser diferentes, según sean estas técnicas más o menos adecuadas.

En primer lugar, se debe tener en cuenta que los hiperparámetros son sensibles a los cambios, ya que controlan el proceso de clasificación de una imagen. Por esto, para optimizar un clasificador, es necesario explorar los espacios de parámetros filtrando aquellos con mayor precisión, que pueden resultar demasiado agoradores y lentos tomando horas o incluso días para encontrar la mejor opción de hiperparámetros que se adapten a su problema.

Por ejemplo, la función detectMultiScale (detección multiescala) tiene dos hiperparámetros que influyen directamente en la clasificación: ScaleFactor y MinNeighbors. El primero determina cuánto se redimensionará la imagen para identificar objetos más pequeños (en este caso, para detección de rostros); este parámetro influye en la detección de personas más alejadas de la cámara, pero con ello, en cambio, la tasa de falsos positivos y la velocidad de detección tienden a aumentar. El segundo parámetro está relacionado con el tamaño del kernel (matriz que evalúa los pixeles vecinos del analizado), lo que hace más clara la imagen, con detalles más relevantes para el clasificador; sin embargo, ello requerirá mayor potencia de procesamiento.

En cuanto al procesamiento de la imagen, cabe mencionar que, aplicando las otras técnicas y filtros que proporciona OpenCV, el rendimiento del clasificador puede ser diferente, por ejemplo, según si se ecualiza o binariza la imagen mediante suavizado gaussiano. Cada una de estas técnicas y filtros cuenta con sus propios hiperparámetros para su optimización.

Otro punto para destacar es la igual importancia de precisión y revocación en la ecuación de puntuación f_1-*score*. Además de ser preciso, el algoritmo debe detectar la mayor cantidad de rostros sin mascarilla protectora en una escena determinada, representada por la medida de desempeño del puntaje f_1-*score* lograda en este proyecto a través del equilibrio entre precisión y revocación. Esto permite buenos resultados al someter el MVP al análisis de un gran flujo de personas.

Una limitación del clasificador Haar Cascade es su restricción para identificar solo caras frontales. En un entorno no controlado, las personas pueden caminar en diferentes direcciones, por lo que no se detectan los rostros que no miran hacia la cámara. Una alternativa para solucionar esta limitación del clasificador es el uso de modelos con técnicas de *deep learning*, que contienen neuronas separadas por capas que, en el transcurso del proceso de entrenamiento, se especializan en el reconocimiento de patrones complejos, como identificar la mascarilla protectora independientemente del ángulo de la cara.

Los datos procesados durante la ejecución del prototipo se almacenan en una base de datos. Este almacenamiento permite cruzarlos con otras fuentes; por ejemplo, datos

públicos puestos a disposición por Datasus, departamento de TI del Sistema Único de Salud de Brasil (SUS). Este cruce de datos puede fortalecer los análisis en el contexto de la pandemia, en relación con la propagación del virus y el uso de mascarillas protectoras.

Conclusión

Considerando la pandemia de COVID-19 y la necesidad de cuidados con respecto a la proliferación del virus, este trabajo presenta una aplicación del uso de técnicas de visión por computadora para monitorear el uso de mascarillas faciales. Se ha desarrollado una aplicación web que permite detectar y registrar imágenes de rostros de personas que no llevan mascarillas, una aplicación que se puede implementar fácilmente en dispositivos de bajo costo.

El sistema pudo detectar rostros que no tenían una mascarilla protectora con una precisión del 63 % y registrar las ocurrencias en una base de datos. Esto permite que los responsables de un determinado entorno, donde se monitorea la aplicación, puedan rastrear la necesidad de implementar medidas eficaces para combatir y prevenir el contagio.

Como propuesta de trabajo futuro, se pretende utilizar técnicas de *deep learning* para incrementar la eficacia de la solución. Tal propuesta debe sopesar los aspectos de precisión facial con respecto a los aspectos de desempeño, de modo que permitan implementar la aplicación en dispositivos de bajo costo.

Agradecimientos

Los autores desean agradecer al Instituto Federal de Educación, Ciencia y Tecnología de São Paulo (IFSP) por su apoyo en el desarrollo de este trabajo. Alexandre Pereira Junior agradece al Programa Institucional de Becas para la Iniciación Científica y Tecnológica del IFSP por su apoyo en la realización de este artículo.

Declaración de divulgación

Los autores declaran que no existe ningún potencial conflicto de interés relacionado con el artículo.

Financiamiento

Los autores declaran que, para la realización de este artículo, han recibido apoyo financiero del Programa Institucional de Becas para la Iniciación Científica y Tecnológica del Instituto Federal de Educación, Ciencia y Tecnología de São Paulo.

Sobre los autores

Alexandre Pereira Junior es estudiante de Tecnología en Análisis y Desarrollo de Sistemas en el Instituto Federal de Educación, Ciencia y Tecnología de São Paulo (IFSP),

en el campus de São Paulo Pirituba. Es becario del Programa Institucional de Becas de Iniciación Científica y Tecnológica del IFSP (PIBIFSP).
https://orcid.org/0000-0002-9438-5858 - Contacto: a.junior@aluno.ifsp.edu.br

Thiago Pedro Donadon Homem es doctor en ingeniería eléctrica por el Centro Universitario FEI, máster en ingeniería eléctrica y licenciado en ciencias de la computación de la Universidade Estadual Paulista - Unesp. Actualmente es profesor en el Instituto Federal de Educación, Ciencia y Tecnología de São Paulo - IFSP, en el campus de São Paulo Pirituba.
https://orcid.org/0000-0003-2140-0129 - Contacto: thiagohomem@ifsp.edu.br

Fabio Oliveira Teixeira es doctor y M. Sc. en el Posgrado en Gestión de la Salud e Informática de la Universidad Federal de São Paulo, y licenciado en ciencias de la computación y gestión de la producción industrial. Es profesor e investigador del Instituto Federal de Educación, Ciencia y Tecnología de São Paulo.
https://orcid.org/0000-0002-1031-2644 - Contacto: fabio.teixeira@ifsp.edu.br

Referencias

Almanza, J. C. (2018). *Visão computacional e aprendizagem automática para aplicações em agropecuária e ciências forenses* [monografía de grado, Universidade Católica Dom Bosco, Campo Grande, Mato Grosso do Sul, Brasil]. http://www.gpec.ucdb.br/pistori/orientacoes/planos/carlos2018.pdf

Araujo Z., L., & Rosito J., C. (2018). Real-time gender detection in the wild using deep neural networks. En *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (pp. 118-125). https://doi.org/10.1109/sibgrapi.2018.00022

Backes, A. R., & Junior, J. J. (2019). *Introdução à visão computacional usando Matlab*. Alta Books Editora.

Botelho, G., Papa, J., & Marana, A. (2018). On the learning of deep local features for robust face spoofing detection. En *2018 31st SIBGRAPI Conference on Graphics, Patterns and Images (SIBGRAPI)* (pp. 258-265). https://doi.org/10.1109/sibgrapi.2018.00040

Duarte, J. C. (2009). *O algoritmo boosting at start e suas aplicacoes* [tesis, Doutor em Ciências-Informática, Pontifícia Universidade Católica do Rio de Janeiro]. Coleção Digital. https://doi.org/10.17771/PUCRio.acad.31451

Chang, A., Wang, A., Mark, A., & Louie, K. (2020). *Documentation—Materialize* [*software*]. Google. https://materializecss.com/

Clark, A., Lundh, F., & GitHub contributors. (2020). *Pillow: Python Imaging Library* (7.2.0) [*software*]. Python. https://python-pillow.org

Data Flair. (2020). *Face Mask Data* [base de datos para descarga]. https://data-flair.training/blogs/download-face-mask-data/

Diário Oficial de São Paulo. (2020, 6 de mayo). *Decreto 59396 2020 de São Paulo SP*. https://bit.ly/3hbhJGbb

Dirolf, M. (2020). *Pymongo: Python driver for MongoDB* (3.11.0) [*software*]. Python. https://pypi.org/project/pymongo/

Finizola, J. S., Targino, J. M., Teodoro, F. G., & Lima, C. A. (2019). Comparative study between Deep Face, Autoencoder and traditional machine learning techniques aiming at biometric facial recognition. En *2019 International Joint Conference on Neural Networks (IJCNN)* (pp. 1-8). https://doi.org/10.1109/IJCNN.2019.8852273

Gehanno, J.-F., Rollin, L., Le Jean, T., Louvel, A., Darmoni, S., & Shaw, W. (2009). Precision and recall of search strategies for identifying studies on return-to-work in Medline. *Journal of Occupational Rehabilitation, 19*(3), 223-230. https://doi.org/10.1007/s10926-009-9177-0

Harmouch, M. (2020, 4 de julio). *Haar cascade classifiers in OpenCV explained visually.* Medium. https://bit.ly/32frGZ4

Hewings-Martin, Y. (2020, 10 de abril). How do SARS and MERS compare with COVID-19? *Medical News Today.* https://bit.ly/2K3gubP

Kissler, S. M., Tedijanto, C., Goldstein, E., Grad, Y. H., & Lipsitch, M. (2020). Projecting the transmission dynamics of SARS-CoV-2 through the postpandemic period. *Science, 368*(6493), 860-868. https://doi.org/10.1126/science.abb5793

Kunz, G. (2019). *Chartist.js Simple responsive charts*. https://gionkunz.github.io/chartist-js/

Lienhart, R., & Maydt, J. (2002). An extended set of Haar-like features for rapid object detection. En *Proceedings. International Conference on Image Processing*. https://doi.org/10.1109/ICIP.2002.1038171

Magdy, W., & Jones, G. J. (2010). PRES: A score metric for evaluating recall-oriented information retrieval applications. En *Proceedings of the 33rd international ACM SIGIR conference on Research and Development in Information Retrieval* (pp. 611-618). https://doi.org/10.1145/1835449.1835551

MongoDB Inc. (2020). *MongoDB Atlas*. https://www.mongodb.com/cloud/atlas

Mordvintsev, A. (2013). *Face detection using Haar cascades.* OpenCV-Python Tutorials. https://bit.ly/3bM9aKX

OpenCV Team. (2020). *OpenCV*. https://opencv.org/

OpenJS Foundation. (2020). *JQuery*. https://jquery.com/

Pallets (s. f.). *Welcome to Flask's documentation* [página web]. Consultado el 14 de agosto de 2020. https://flask.palletsprojects.com/en/1.1.x/

Pontes, R. (2011). *Inteligência artificial nos investimentos*. Clube de Autores (managed).

Portal do Governo. (2020, 9 de abril). *Governo de SP apresenta Sistema de Monitoramento Inteligente contra coronavírus*. Governo do Estado de São Paulo. https://bit.ly/2X3ReFn

Python Software Foundation. (2020a). Base64—Base16, Base32, Base64, Base85 Data Encodings (3.8.5). *Documentation*. Consultado el 14 de agosto. https://docs.python.org/3/library/base64.html

Python Software Foundation. (2020b). Io—Core tools for working with streams (3.8.5). *Documentation*. Consultado el 14 de agosto. https://docs.python.org/3/library/io.html

Reis, F. (2020, 7 de abril). Estudo avalia eficácia das máscaras de proteção no combate ao coronavírus. *Pfarma*. https://pfarma.com.br/coronavirus/5399-eficacia-mascaras.html

Sistema de Monitoramento Inteligente do Governo de São Paulo (SIMI-SP). (2020). *Isolamento.* Governo do Estado de São Paulo. https://www.saopaulo.sp.gov.br/coronavirus/isolamento/

Vieira, J., Ricardo, O., Hannas, C., Kanadani, T., Prata, T., & Kanadani, F. (2020). What do we know about COVID-19? A review article. *Revista Da Associação Médica Brasileira, 66*(4), 534-540. https://doi.org/10.1590/1806-9282.66.4.534

Viola, P., & Jones, M. (2001). Rapid object detection using a boosted cascade of simple features. En *Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition. CVPR 2001*. https://doi.org/10.1109/CVPR.2001.990517