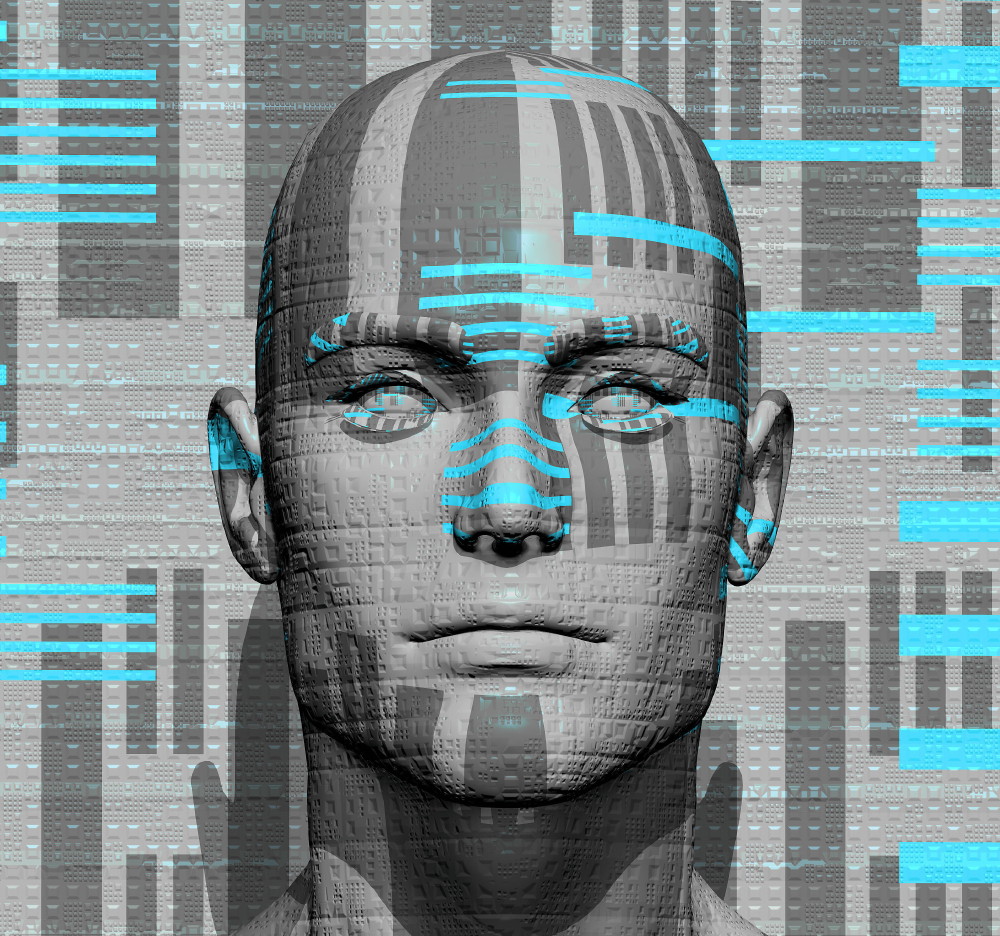
**Visión Por Computador**

**Reconocimiento de Caras**



**Realizado por:**

Francisco Pérez Hernández

Cristina Zuheros Montes

1 Introducción

El objetivo de nuestro trabajo es realizar una técnica con la que podamos reconocer caras en una fotografía.

Para realizar esta tarea lo primero que hemos conseguido es una base de datos de rostros [1]. Estos rostros tienen una limitación, cada fotografía es de un solo rostro en distintas condiciones de iluminación.

Algunos problemas que presuponemos de inicio son:

* Nuestro algoritmo tendrá distintos resultados en función de la iluminación.
* Los rostros no deberían salir en otra posición que no sea vertical.

Por lo tanto, y viendo estos inconvenientes, vamos a proponer una serie de técnicas como son:

* Pasar imágenes en RGB, de color carne a blanco-negro.
* Pasar imágenes en RGB a YCrCb, con lo que obtenemos la piel en las imágenes.

2 Pasar de color carne a blanco-negro

La primera idea que puede saltar cuando piensas como debería actuar el algoritmo, sería sacar el color carne de una persona y a partir de ahí obtendríamos el cuerpo de esta. Esto supone una limitación, y es que las personas con un tono de piel que no sea color carne (dentro de unos rangos), no serán localizadas, como personas con un tono más amarillo, o marrón.

Sacada esta idea, se ha buscado cual podría ser el tono color carne medio, para poder tener un algoritmo que tenga buenos resultados. Para ello también tendremos una tolerancia, para que el rango de colores pueda ser más amplio.

Por lo tanto, nuestro algoritmo será:

Vector<Mat> DetectarRosa(vector<Mat> imágenes\_caras, int tolerancia);

Por lo tanto, nuestro algoritmo recibirá un vector con imágenes y una tolerancia determinada. Este nos devolverá un vector con las imágenes pasadas a blanco-negro.

Tras diferentes pruebas se ha visto que los valores de RGB, que mejor función, junto con la tolerancia, son los siguientes:

R=253, G=221, B=201, tolerancia=70

Para estos valores, algunos de los resultados obtenidos son:









Como se puede ver en las imágenes, los resultados de este algoritmo son muy malos ya que en función de la iluminación, podría reconocer color carne en zonas donde no las hay.

En conclusión, este algoritmo no es lo suficiente bueno para poder trabajar sobre el. Por lo tanto, en el siguiente apartado vamos a implementar otra técnica, con la cual vamos a convertir la imagen RGB a YCrCb para poder quitar iluminación de las imágenes.

3 Sacar piel de las imágenes

Como se ha visto en el apartado anterior, sacar el color carne en una imagen obtiene muy malos resultados. Por lo tanto, vamos a pasar a detectar la piel de una imagen.

Para ello nos hemos basado en este paper [2], en el cual usa esta idea y vamos a llevarla a cabo.

3.1 Sacar piel

Primeramente, creamos la función:

Mat TransformarDeRGBAYCrCbYPasoABlancoNegro(Mat, valY, minCr, maxCr, minCb, maxCb);

Con esta función, vamos a aplicar un filtro a la imagen en RGB para quitar la iluminación. Como pensamos que podríamos modificar algo de la transformación que hacia OpenCV, la hemos implementado nosotros para poder controlarla en caso de no tener resultados del todo positivos.

Lo primero que hacemos es transformar la imagen de RGB a YCrCb con un método auxiliar que llamamos RGBtoYCrCb (lo explicaremos a continuación). Una vez que tenemos la imagen en YCrCb vamos a buscar la piel de acuerdo a los valores Y, Cr, Cb. Establecemos las cotas según hemos ido viendo que los resultados eran mejores o peores. Aquellos pixeles que tengan unos valores de Y, Cr y Cb dentro del umbral que imponemos (tonos de piel) vamos a pasarlos a color blanco y los que estén fuera los pasaremos a color negro. De este modo tendremos la imagen en tonos blancos y negros donde el blanco representa la piel de la imagen.

Veamos el método Mat RGBtoYCrCb(Mat im\_original);

Creamos la matriz de transformación :

matriztra.at<double>(0, 0) = 0.3;

matriztra.at<double>(1, 0) = 0.6;

matriztra.at<double>(2, 0) = 0.1;

matriztra.at<double>(0, 1) = -0.2;

matriztra.at<double>(1, 1) = -0.3;

matriztra.at<double>(2, 1) = 0.5;

matriztra.at<double>(0, 2) = 0.5;

matriztra.at<double>(1, 2) = -0.4;

matriztra.at<double>(2, 2) = -0.1;

Para pasar de RGB a YcrCb, multiplicamos el vectr RGB por la matriz de transformación. Hemos comparamos los resultados con el método que ofrece OpenCV (ConverTo(RGBtoYCrCb)) y hemos visto que son exactamente iguales a los que obtenemos.

3.2 Recortar piel

196 imágenes malas de 452, 43,3628%

3.3 Verificar si hay ojos en la cara

no en 82 de 256, 32,0313%

X-1 Otras pruebas

gauss

X Referencias

[1] – Base de datos de caras:

[2] – Paper sobre sacar piel:

[3] -