Práctica 2: Corrección de Ojos Rojos

Antonio José Moya Díaz

30 de abril de 2013

INTRODUCCIÓN.- Es habitual, cuando se retratan personas o animales usando flash, la aparición del efecto conocido como 'ojos rojos'. El uso del flash suele ser habitual en condiciones de escasa luz donde la pupila del retratado suele estar dilatada, lo que provoca que la propia luz del flash pase a través de ésta reflejándose en la retina y devolviendo a la cámara el clásico efecto de unas pupilas de un color rojo intenso e irreal.

La mayoría de las cámaras modernas poseen métodos de reducción de este efecto, normalmente, basados en la física del ojo humano. No obstante, con relativa frecuencia estos métodos son insuficientes o puede darse el caso de no disponer de una cámara con esta opción, por lo que acaba siendo necesario un tratamiento digital de la imagen para la corrección de los ojos rojos.

En este documento se pretende mostrar un algoritmo automático de corrección del efecto de ojos rojos.

1. Detección del área afectada

El primer paso para la corrección del efecto de ojos rojos es localizar la zona afectada sobre la que aplicar la corrección que deseemos.

Si queremos un algoritmo realmente automático hemos de abordar el problema de una detección sin que el usuario se vea obligado a señalar la posición del ojo.

Observando el ojo rojo

Para abordar la detección automática debemos ser capaces de 'decirle' al ordenador lo que de manera subjetiva para un observador es evidente. Por tanto, lo primero que deberíamos preguntarnos es, nivel de imagen, de píxeles, ¿qué es un ojo rojo?

Figura 1: Efecto de los ojos rojos

Pues a poco que sepamos algo sobre imágenes seremos capaces de acertar a decir que un ojo rojo es un área que presenta un fuerte tono de color rojo, con una altísima saturación. Tono y saturación. ¿sabe un ordenador qué es el tono y la saturación?

El espacio de color HSV

Las imágenes captadas por una cámara de fotos suelen estar en el espacio de color conocido como RGB (Red, Green and Blue), que supone un buen modelo para representar imágenes dada la fisiología del ojo humano y de cómo las pantallas de los aparatos están preparadas para ello. Sin embargo, con dicho modelo resulta muy difícil caracterizar algunos parámetros de la imagen como son los previamente citados tono y saturación. Para ello se introduce el espacio de color HSV.

El espacio de color HSV (Hue -tono-, Saturation and Value) codifica las imágenes en base a otros parámetros como son el tono, la saturación y un valor asociado con el brillo. Así que ya tenemos parametrizados en el ordenador el tono y la saturación que nos hacen falta.

Obtención de la máscara

Como ya adelantamos anteriormente, el área que buscamos de los ojos rojos será una zona de la ima-

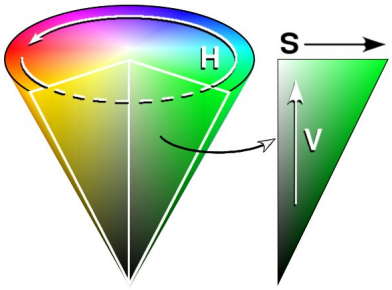

Figura 2: Espacio de color HSV

gen con un valor de tono en torno, y muy próximo, al rojo. Además, será un área con un considerable valor en saturación.

Para la detección del color rojo hemos de conocer el espacio de color HSV con un poco de profundidad. Este espacio de color suele representarse como un cilindro o un cono (figura 2) donde el valor del tono se representa como un ángulo de giro estando el color rojo puro en 0º, o equivalentemente en 360º. Así pues debemos establecer un intervalo en torno a dicho ángulo que será el umbral sobre el que buscaremos valores de rojo.

Tras varias reiteraciones y pruebas en base a las imágenes dadas, se ha determinado que los píxeles de color rojo se hallarán en el intervalo [-36º, 7.2º]. MATLAB usa el espacio HSV con el valor del tono normalizado, correspondiendo el valor 0 a 0º y el valor 1 a 360º. Así pues, se ha recorrido la imagen en busca de píxeles que cumplan:

$$I_H(x,y) \in [0,0{,}02] \cup [0{,}9,1] \tag{1}$$

De forma similar se han probado diversos valores de saturación encontrando que se tratan éstos de unos valores considerablemente más laxos. Si se escogían unos valores muy estrictos, muchos píxeles pertenecientes al área afectada por los ojos rojos quedaban fuera de la máscara. Así preferimos bajar el nivel de saturación hasta niveles donde no quedasen píxeles de ojo rojo fuera de la máscara, aunque esto nos hiciese entrar en la máscara bastante falsos positivos, con la esperanza de poder eliminar muchos de éstos a posteriori. Finalmente se optó por:

$$I_S(x,y) \in [0{,}6,1] \tag{2}$$

El resultado de esta primera máscara puede observarse en la figura 3 dónde, aparte de los ojos, se hacen patentes una buena cantidad de falsos positivos.

Eliminación de falsos positivos puntuales

Para la corrección de esos numerosos pequeños falsos positivos vamos a usar técnicas de morfología matemática.

Se ha optado por un proceso de apertura, ésto es, de una erosión seguida de una dilatación. El paso de la erosión nos consigue eliminar la gran mayoría de los falsos positivos, pero deja las áreas de interés ligeramente más pequeñas de lo que nos interesa, por eso, un posterior proceso de dilatación agrandará las áreas de interés dejándolas de un tamaño similar al que tenían antes de la erosión.

Sin embargo, tras la apertura, aún podría ocurrir que las áreas de interés fuesen ligeramente más pequeñas de lo que nos interesa. Es más, con el fin de corregir con la mayor precisión el efecto de los ojos rojos, y más teniendo en cuenta la corrección que veremos más adelante se ha realizado, resulta mas interesante que la imprecisión de la máscara sea por exceso y no por defecto. Es por ello que volvemos a someter la máscara a un segundo proceso de dilatación.

En ambos casos se ha usado un kernel circular de radio 2, y los resultados podemos observarlos en la figura 4.

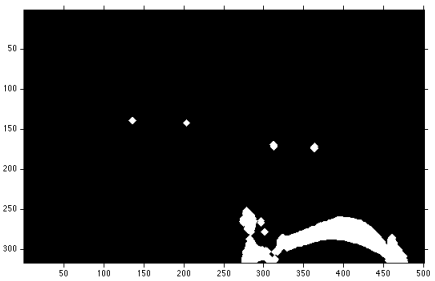

Figura 4: Máscara mejorada con morfología

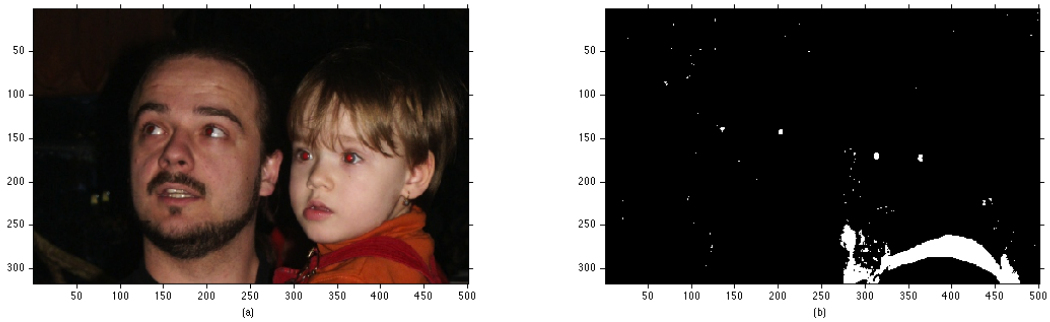

Figura 3: (a) Imagen original, (b) Máscara previa con falsos positivos

Eliminación de falsos positivos de gran extensión

Sin embargo, como se puede observar en la figura 4, aquellas zonas cuya extensión o área sea comparable o mayor al área de los ojos, sobrevivirán a las operaciones morfológicas. Debemos, pues, buscar alguna forma de poder detectarlas y borrarlas.

Para ello vamos a usar algunas de las funciones que nos proporciona MATLAB, como por ejemplo *bwlabel* que nos etiqueta todos los elementos conectados de la imagen y *regioprops* que nos calculará algunas propiedades básicas de esos elementos etiquetados.

Hemos de tener en consideración que los ojos serán, normalmente, un área más bien pequeña de la imagen. Por lo que una posible solución sería comparar el área de los ojos con el área de otras zonas seleccionadas por la máscara y quedarnos con las más pequeñas confiando en que éstas se traten de los ojos.

Para ello usamos la función *regionprops* y extraemos el área de las figuras. La condición, como ha sido anteriormente expresada, no es la más óptima que uno pudiera considerar. Pudieran haberse detectado erróneamente zonas más pequeñas que las de los ojos, por lo que quedarse con las más pequeñas de cuantas han sido detectadas podría ser contraproducente y acabar eliminando, precisamente, la zona de los ojos.

La primera comprobación que se propone es contar cuantos elementos conectados hay, pues si solo hay 2 es razonable pensar que éstos serán los ojos y no será necesario hacer ningún procesado de poda de regiones indeseadas. En caso de que haya más de 2 elementos conectados, y considerando el caso anteriormente expuesto de que las zonas indeseadas sean menores o del mismo orden que los ojos rojos, se propone descartar aquellas áreas que cumplan lo siguiente:

$$A > 2\bar{A} \tag{3}$$

Donde A es el área del elemento conectado y $\bar{A}$ la media de las áreas de todos los elementos conectados.

Aplicando esta condición a la máscara anterior obtenemos la máscara final considerada que podemos observar en la imagen **??** donde ya solo se aprecian las áreas de los ojos rojos.

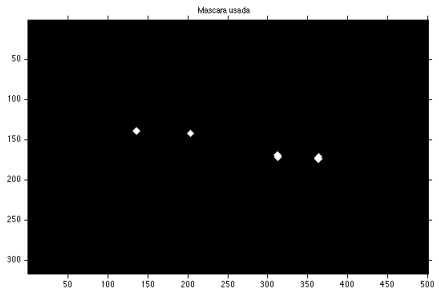

Figura 5: Máscara final considerada

2. Corrección del color

Una vez detectada el área afectada por los ojos rojos, hemos de corregir el defecto. Para la corrección se ha optado por un método bastante sencillo

[1]. Trabajando en el espacio de color RGB se basa en cambiar la componente de la banda roja, para ese área, por un nivel 'acromático' calculado como la media de los niveles verde y azul.

Con un procedimiento tan sencillo podemos obtener resultados tan buenos como los que se pueden apreciar en la imagen.

3. Resultados

Para evaluar los resultados del algoritmo propuesto se ha intentado elaborar un índice lo más objetivo posible. Para ello se han considerados dos casos distintos a los que se les ha aplicado, sin embargo, el mismo procedimiento.

El índice de calidad se propone como sigue:

$$indice = \frac{A_{err}}{A_{total} - A_{err}} \tag{4}$$

Donde A_{err} corresponde al área ocupada por el error, y A_{total} el área total detectada por la máscara. Tal y como se ha definido vemos que si no existe error alguno el valor del mismo será 0. Ocurre además que si el área errónea es mayor que el área de los ojos rojos, es decir, si $A_{err} > A_{total} - A_{err}$ entonces el índice será mayor que 1, y peor cuanto mayor sea su valor. Si por contra el área erronea es menor que el área de los ojos rojos el valor del índice será menor que 1 y mejor cuanto más próxima a cero, y podremos considerar que se ha realizado una corrección, no perfecta, pero si razonablemente buena.

Figura 6: Imagen corregida

Falsos positivos

Para el cálculo de los falsos positivos se ha tomado como área errónea todas aquellas áreas que aparezcan en la máscara sin pertenecer a los ojos rojos.

Podemos ver los resultados en la tabla 1 donde se ha calculado del índice para la batería de imágenes dada. Se comprueba el bajo índice, en general, de falsos positivos.

Imagen	Índice
ojo1.jpg	10.56
ojo2.jpg	0
ojo3.jpg	0
ojo4.jpg	0
ojo5.jpg	0
ojo6.jpg	0.64
ojo7.jpg	0
ojo8.jpg	0

Cuadro 1: Resultado para falsos positivos

Imagen	Índice
ojo1.jpg	0.08
ojo2.jpg	0.5277
ojo3.jpg	0
ojo4.jpg	0.4915
ojo5.jpg	0
ojo6.jpg	0
ojo7.jpg	0
ojo8.jpg	0

Cuadro 2: Resultado para falsos negativos

Falsos negativos

El cálculo de los falsos negativos ha sido más delicado y la objetividad de los resultados debe ser tenida en cuenta con pinzas. Para ésto, en aquellas imágenes en las que se observaba que el algoritmo no corregía completamente el defecto de los ojos rojos, se ha seleccionado a mano, intentando ser lo más preciso posible, el área que debería haber sido corregida idealmente. Luego se ha intersecado con la máscara de los ojos rojos obteniendo la diferencia entre ambas, y

esa diferencia es la que se considera en este caso A_{err}. Los resultados se pueden ver en la tabla 2.

Hay que hacer una breve consideración respecto a los índices. Éstos se han calculado en base a suma de áreas totales. Quiere decir que si tenemos muchos puntos dispersos pero pequeños, podríamos obtener un resultado de alto índice de error según éste índice y, sin embargo, perceptualmente tener una imagen con una corrección casi perfecta. También se puede dar, obviamente, el resultado a la inversa, un área pequeña pero concentrada en un solo punto podría hacerse claramente visible.

Referencias

[1] B. Smolka, K. Czubin, J.Y. Hardeberg, and others *Towards an automatic redeye effect removal*. Elsevier Science B.V. 2002