

Efectos

Visión por Computador

Alejandro Paricio García, 761783
Fernando Peña Bes, 756012

Universidad de Zaragoza
28 de julio de 2021

Índice

1. Introducción	1
2. Contraste	1
2.1. Ajuste del brillo y contraste de la imagen	1
2.2. Ecualización del histograma	2
2.3. CLAHE	2
3. Alien	3
4. Posterize	3
5. Distorsión de barril y de cojín	4
6. Thresholding	5
7. Twirl	5
8. DuoTone	7
9. Interfaz gráfica	7
Referencias	9

1. Introducción

En la primera práctica de la asignatura se ha desarrollado una aplicación sencilla con una interfaz gráfica que permite seleccionar y modificar el filtro aplicado en cada momento, mostrando los resultados en tiempo real. Siguiendo las indicaciones del enunciado, se ha tratado de obtener una aplicación usable. En el vídeo adjunto a la memoria se demuestra el uso y resultados con cada filtro obtenidos en tiempo real.

2. Contraste

2.1. Ajuste del brillo y contraste de la imagen

Con el objetivo de crear un filtro que permita modificar el brillo y el contraste de una imagen se ha implementado de manera acorde a lo visto en las clases de la asignatura, multiplicando cada píxel por la ganancia para aumentar el contraste y sumándole un sesgo para modificar el brillo. El filtro se puede definir con la ecuación

$$g(\mathbf{x}) = af(\mathbf{x}) + b,$$

donde a es la ganancia y b el sesgo.

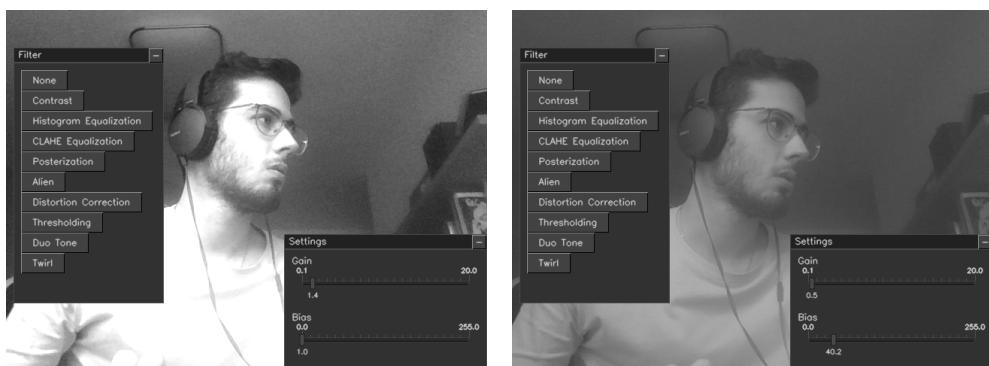


Figura 1: Ajuste brillo y contraste

2.2. Ecualización del histograma

La ecualización del histograma busca redistribuir y modificar la intensidad de cada píxel para que el histograma acumulativo sea lineal. OpenCV proporciona una función que lleva a cabo este proceso a partir de una imagen en escala de grises, *equalizeHist*. Se ha recurrido a ella para obtener las imágenes a continuación:

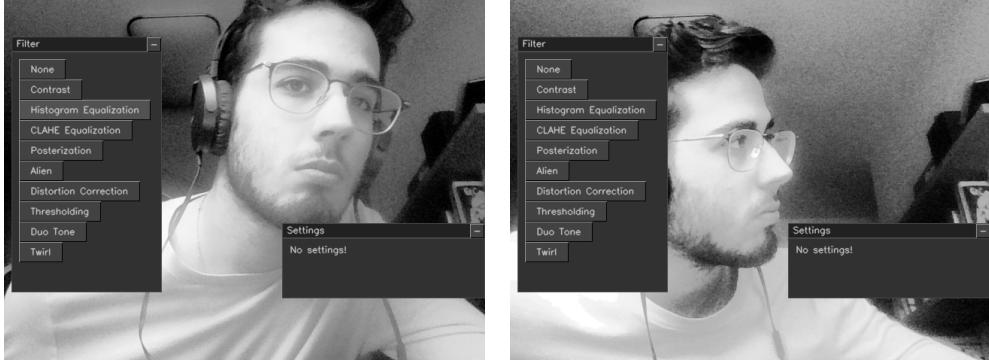


Figura 2: Ecualización del histograma

2.3. CLAHE

Adicionalmente, se ha probado otro tipo de ecualización del histograma adaptativo que tiene en cuenta a la hora de modificar la intensidad de un píxel únicamente los píxeles vecinos y que, además, limita la variación introducida en el histograma original [1]. También está implementado en OpenCV, por lo que se ha aprovechado esta herramienta.

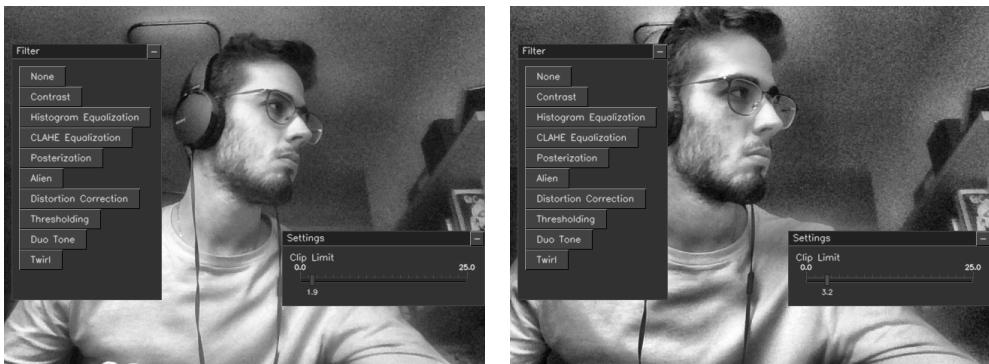


Figura 3: CLAHE

3. Alien

El filtro de alien supone la detección de las partes de la imagen que se corresponden con piel humana, para una vez detectadas sustituirlas por el color verde. Para ello, se han obtenido dos máscaras diferentes en función de los rangos en los 3 canales de la imagen. La primera de ellas se obtiene sobre la imagen en modelo de color HSV, con los valores inferiores para cada canal de [0, 40, 60] y superiores [20, 150, 255] tras lo que se aplica un *erode* y un *dilate* con un kernel elíptico, siguiendo el ejemplo proporcionado por [2]. La segunda se obtiene con la imagen en el modelo de color YCbCr, con valores inferiores [0, 138, 67] y superiores [255, 173, 133], continuando la idea observada en múltiples ejemplos como [3]. Finalmente, se combinan las dos máscaras y se pueden obtener resultados como los siguientes:

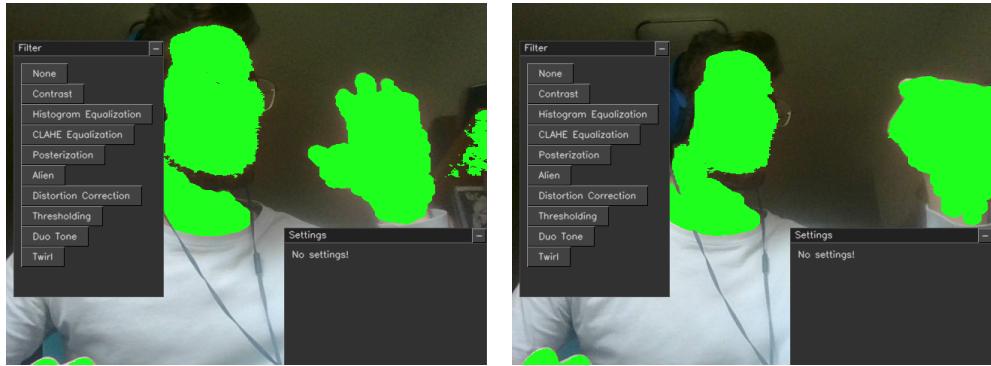


Figura 4: Efecto Alien

4. Posterize

El filtro Posterize tiene como objetivo reducir el número de colores de la imagen, produciendo resultados especialmente agradables a la vista. Para su implementación se ha decidido dividir el espacio de color RGB en tantos fragmentos por canal como colores por canal se deseen en la imagen resultante. Para ello cada valor de cada canal se modifica para dejarlo con el valor de uno de los fragmentos seleccionados. Los resultados obtenidos son los siguientes:

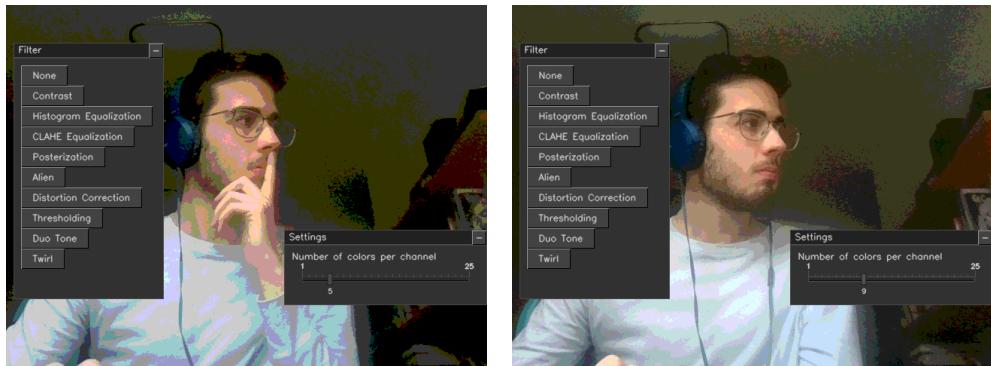


Figura 5: Efecto Posterize

5. Distorsión de barril y de cojín

Otro de los filtros exigidos por el enunciado era aplicar una distorsión de barril y de cojín. Para ello, se ha empleado una fórmula que mapea cada píxel de la imagen en otra posición, junto a una función de remap para efectuar el *inverse mapping*.

cy = centro de la imagen en el eje X

cx = centro de la imagen en el eje y

$$r^2 = (x - cx) * (x - cx) + (y - cy) * (y - cy)$$

$$x = (x - cx) * (1 + k1 * r^2 + k2 * r^4)$$

$$y = (y - cy) * (1 + k1 * r^2 + k2 * r^4)$$

Sumar cx a cada coordenada x , y cy a cada coordenada y

Dando distintos valores a $k1$ y $k2$ tanto positivos como negativos se puede obtener tanto la distorsión de cojín como la de barril.

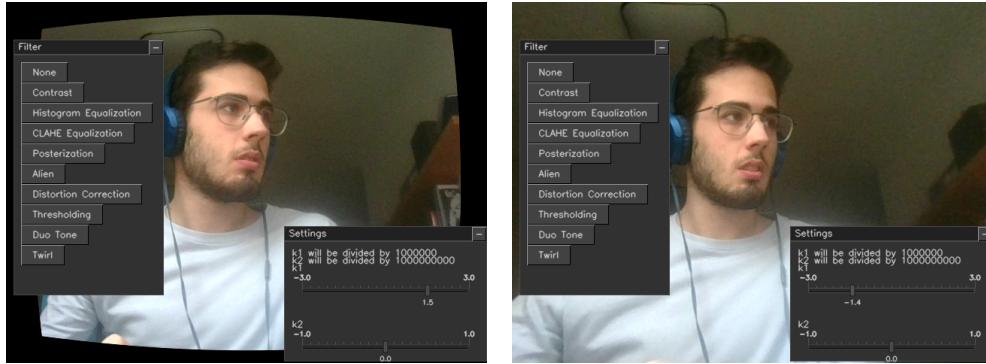


Figura 6: Distorsión de barril y de cojín respectivamente

6. Thresholding

Otro filtro implementado consiste en una umbralización. Se basa en elegir un valor umbral, asignando a aquellos píxeles con un valor mayor (en la imagen en escala de grises) el color blanco y a aquellos menores el negro. Con ello se consigue transformar la imagen a únicamente blanco (255) y negro (0). Además, se proporciona en la aplicación generada la posibilidad de invertir la asignación de blanco o negro de los píxeles. Para la implementación se ha utilizado la función *threshold* de OpenCV.

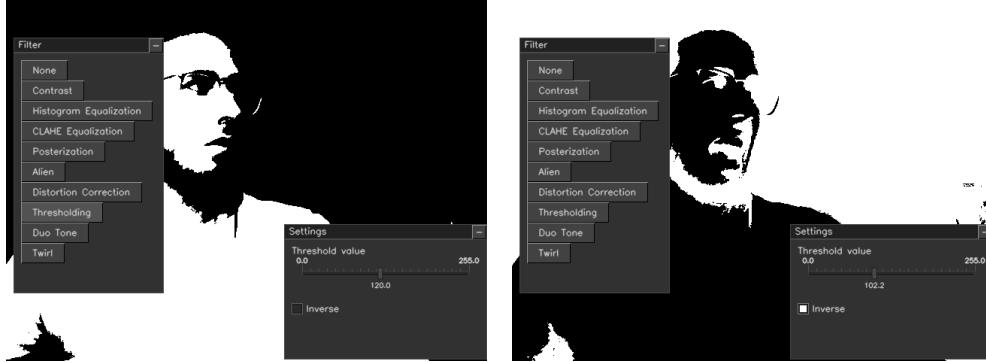


Figura 7: Efecto threshold y threshold inverso

7. Twirl

Este efecto crea un remolino en la imagen, distorsionando de forma circular la zona en la que se encuentra. El remolino se define mediante su radio, centro y cantidad de deformación en espiral. El centro se puede modificar haciendo clic derecho sobre un punto de la imagen en tiempo real.

Para conseguir este efecto, se realiza un mapeo inverso de los píxeles afectados por el remolino. Para cada píxel de la imagen final que se encuentra dentro del área del remolino se calcula cuál es el píxel que le corresponde en la imagen inicial. Los píxeles fuera de este área no son modificados.

A continuación se detalla cómo se busca el píxel de la imagen original que se mapea en una cierta posición de la imagen final, el método está basado en [4]. Vamos a denotar con (i', j') la posición de un píxel en la imagen final, de tamaño $w \times h$, que se encuentra dentro del área del remolino. El centro del remolino es (cx, cy) , tiene radio r y su cantidad de giro es $twist$. Para comenzar, se calcula la posición del píxel respecto al centro del remolino:

$$\begin{aligned} x_p &= j' - cx \\ y_p &= i' - cy, \end{aligned}$$

y su distancia al mismo

$$d = \sqrt{x_p^2 + y_p^2}.$$

Los píxeles se rotan alrededor del centro del remolino, con un ángulo de rotación θ proporcional a la cantidad de giro del remolino, pero inversamente proporcional a la distancia del punto con el centro:

$$c = 1 - (d/r)$$

$$\theta = \text{atan}2(y_p, x_p) \cdot \text{twist} \cdot c \cdot 2\pi.$$

La posición del píxel de la imagen original que nos interesa respecto al centro es

$$i_t = \sin(\theta) \cdot d + y$$

$$j_t = \cos(\theta) \cdot d + x,$$

y su posición en la imagen original,

$$(i, j) = (\min(w - 1, \max(0, i_t)), \min(h - 1, \max(0, j_t))).$$

Si la posición calculada está fuera de los límites de la imagen, se toman los píxeles del borde. Una vez calculada la posición de este píxel, se copia en la posición (i', j') de la imagen final.

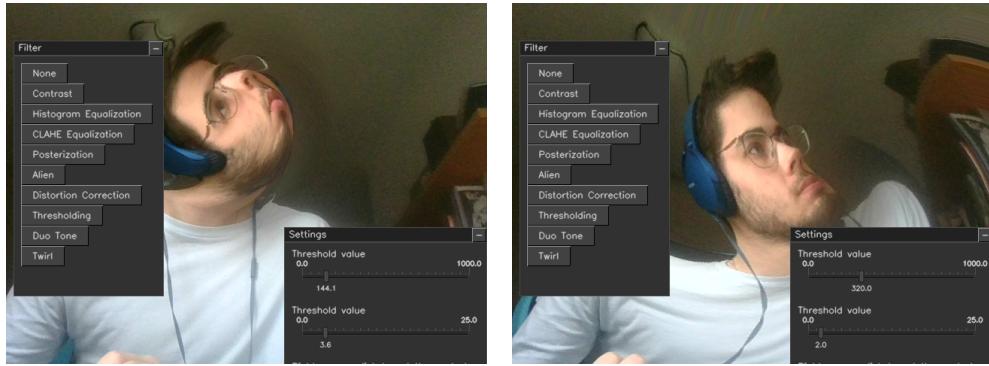


Figura 8: Efecto twirl

8. DuoTone

Finalmente, se ha encontrado un filtro que permite representar imágenes con dos tintas, una técnica que se usaba antiguamente cuando imprimir a todo color con 4 tintas (CMYK) era muy caro. Su implementación ha seguido el tutorial de [5]. Las posibilidades de este efecto se aprecian mejor en el vídeo adjunto.

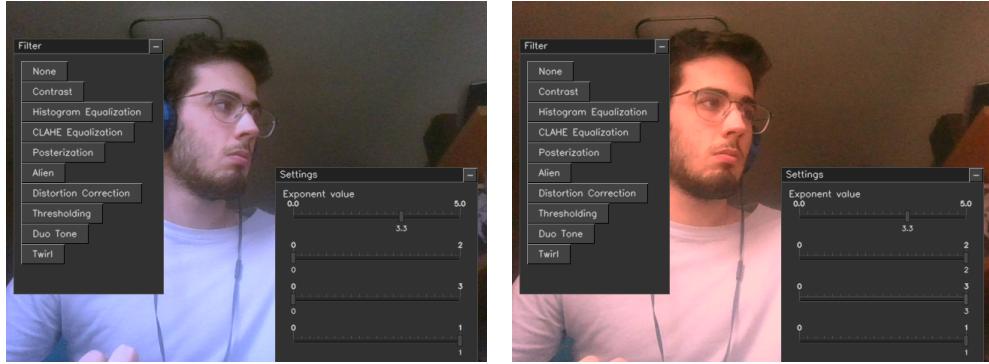


Figura 9: Efecto Duo Tone

9. Interfaz gráfica

Como se puede ver en las imágenes anteriores se ha creado una sencilla interfaz gráfica para poder seleccionar los efectos y modificar sus parámetros en tiempo real. Se ha utilizado la librería cvui [6], que implementa una GUI muy sencilla sobre las primitivas de OpenCV, sin hacer uso de dependencias externas.

La librería consta únicamente de una cabecera, `cvui.h`. Adicionalmente se ha usado la clase `EnhancedWindow`, disponible también en el repositorio de cvui, para crear las ventanas flotantes que se pueden mover por la imagen y minimizar.

Por último, se ha añadido la opción de cargar imágenes de disco para poder aplicarles efectos, llamando al programa con la ruta de una imagen como argumento, y la posibilidad de guardar las imágenes modificadas presionando la tecla `c` del teclado.

En la siguiente figura se muestra el resultado de modificar una imagen cualquiera y capturarla pulsando `c`.



(a) Aplicación de un filtro sobre una imagen



(b) Imagen capturada al pulsar c

Figura 10: Modificación y guardado de imágenes en disco

Referencias

- [1] Adaptative histogram equalization. https://en.wikipedia.org/wiki/Adaptive_histogram_equalization. Consultado el 28/02/2021.
- [2] Skin detection: A step-by-step example using python and opencv. <https://github.com/Jeanvit/PySkinDetection/blob/master/src/jeanCV.py>. Consultado el 28/02/2021.
- [3] Skindetection. <https://github.com/CHEREF-Mehdi/SkinDetection>. Consultado el 28/02/2021.
- [4] Tutorial OpenCV (filtro twirl). <http://acodigo.blogspot.com/2017/01/tutorial-opencv-filtro-twirl.html>. Consultado el 03/03/2021.
- [5] Photoshop filters in OpenCV. <https://learnopencv.com/photoshop-filters-in-opencv/>. Consultado el 28/02/2021.
- [6] Librería cvui. <https://github.com/Dovyski/cvui>. Consultado el 03/03/2021.