VISION POR COMPUTADOR

Práctica 1

Arón Collados (626558)
Cristian Román (646564)

Efecto contraste:

Para modificar el contraste se ha utilizado una sencilla función con cada pixel.

x= alfa *x + beta

Siendo alfa el factor de contraste, cuyo rango aceptable de valores es [1.0-3]. Se ha testado la función en las escalas RGB y HSV, siendo esta última la que mejores resultados ha dado.

Ilustración 1: Efecto contrastre RGB

Ilustración 2: Efecto contrastre HSV

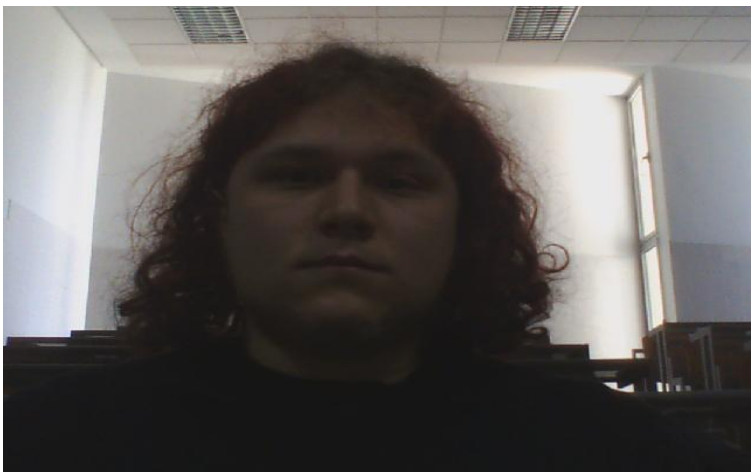

Ilustración 3: Foto original

Ecualización de histograma:

Se han utilizado las funciones proporcionadas en los tutoriales de la librería OpenCV, para lograr este efecto.

```
split(imagen, rgb );
```

Separa los canales RGB de la imagen.

```
equalizeHist(rgb[0], rgb[0]);
equalizeHist(rgb[1], rgb[1]);
equalizeHist(rgb[2], rgb[2]);
```

Ecualiza cada uno de los niveles por separado.

```
merge(rgb,nuevaImagen);
```

Une las capas en una nueva imagen.

Ilustración 4: Imagen tras ecualizar su histograma

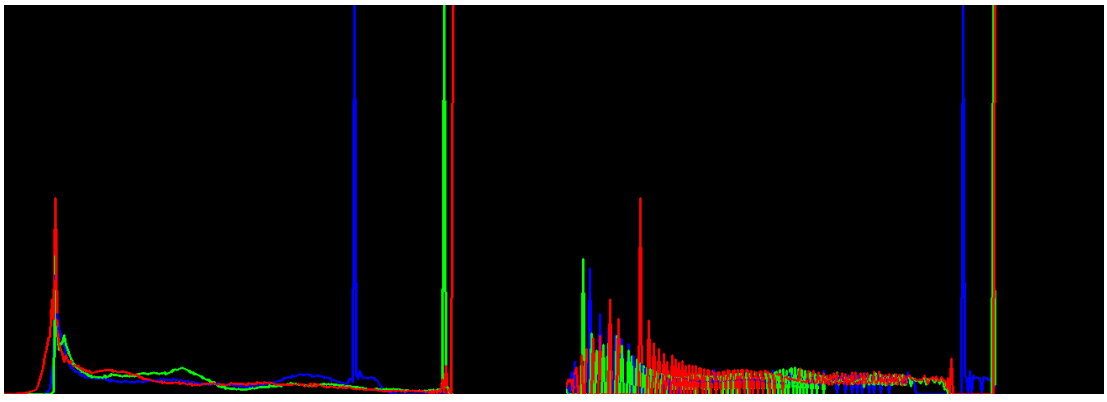

Ilustración 5: Histograma sin ecualizar

Ilustración 6: Histograma ecualizado

Efecto alien:

Para la detección de la piel se ha usado rangos en RGB, HSV y YCrCb. Se han probado los tres métodos, tanto por separado como conjuntamente.

Ilustración 7: Usando los RGB, HSV y YCrCb

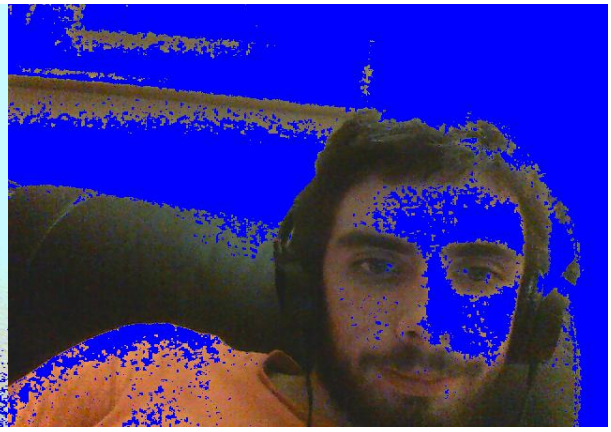

Ilustración 8: Usando RGB

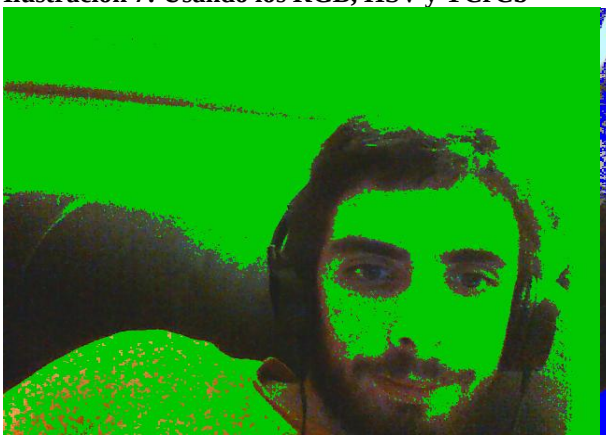

Ilustración 9: Usando YCrCb

Ilustración 10: Usando HSV

Efecto póster:

Para este efecto se ha utilizado una fórmula extraída de G. Bradski and A. Kaehler: Learning OpenCV: Computer Vision with the OpenCV Library, O'Reilly, 2008.

```
x =  (x / div * div + div / 2)
```

Haciendo una comparativa se puede observar que la escala HVS se asemeja más a la imagen original. Y que la escala RGB genera colores artificiales que no existían en la imagen original.

Ilustración 11: Efecto póster con RGB

Ilustración 12: Efecto póster con HSV

Efecto distorsión

Para realizar este efecto se ha aplicado la fórmula:

$$x=(tx * (1 + kx * r^2) * correctorX + Cx)$$
$$y=(ty * (1 + ky * r^2) * correctorY + Cy)$$

x=posición del pixel en x
y=posición del pixel en y
tx=distancia del pixel en el eje x a la mitad de la anchura de la imagen en valores absolutos
ty=distancia del pixel en el eje y a la mitad de la altura de la imagen en valores absolutos
r=distancia desde la posición actual al centro de la imagen
C=coeficientes
Corrector=valores precalculados para que la imagen ocupe el total de la resolución de la imagen

Si se usa un coeficiente positivo se obtiene una distorsión de barril, mientras que si es negativo, se obtiene una distorsión de cojín.

Ilustración 14: Efecto barril

Ilustración 13: Efecto cojín

Efecto invertir:

Se ha realizado un reubicación de los pixeles en la imagen, usando la función proporcionada en los tutoriales.

```
remap(imagen, dst, mapx, mapy, CV_INTER_LINEAR, BORDER_CONSTANT, Scalar(0, 0, 0));
```

Efecto binario:

Convierte en negro o en blanco el pixel dependiendo del brillo del mismo. De este modo si el brillo total es mayor a (255/2)*3 se convertirá en negro en caso contrario sería blanco.

Efecto negativo:

Se ha sustituido cada pixel de la imagen por su equivalencia inversa. Aplicando una simple transformación.

x = 255 - x

Ilustración 16: Efecto negativo

Reducción de ruido mediante filtros:

Se ha intentado reducir el de la imagen con dos filtros distintos, el gaussiano y el binomial. El filtro gaussiano es más potente pero una buena opción para imágenes en tiempo real, porque es costoso en calculo y añade cierto retraso a la imagen.

1	4	6	4	1
4	16	24	16	4
6	24	36	24	6
4	16	24	16	4
1	4	6	4	1

1	2	1
2	4	2
1	2	1

Ilustración 18: Imagen con ruido

Ilustración 17: Imagen corregida

Fuente de la Imagen: Wikipedia