Detección de piel con Matlab

Alejandro Amat,
Pablo Díaz
Profesor: Oscar Pina

***Resumen*—En este documento se presenta la primera parte del trabajo del laboratorio de PIV relacionada con la detección de píxeles de piel utilizando el software Matlab realizada por los alumnos Alejandro Amat y Pablo Díaz.**

I. Introducción

El trabajo a presentar consiste en detectar píxeles de piel de imágenes entrantes y, por lo tanto, clasificarlos en función de si se consideran piel o no.

$$\begin{aligned} H_o &= x_{i,j} \in P \\ H_1 &= x_{i,j} \notin P \end{aligned} \tag{1}$$

Como indica (1) consideramos la hipótesis inicial como la detección de un píxel de piel. Siendo P, el conjunto de estos. Nótese que en la expresión nos refermos a i,j como los índices de los píxeles que tomarán valores en función del ancho y longitud de cada imagen; Dado que no todas tienen el mismo tamaño.

Para poder discernir entre ambas hipótesis primero debemos crear un modelo estadístico que nos permita comprobar y comparar los valores obtenidos. Es por eso que contamos con un total de 101 imágenes de entrenamiento para poder establecer una estadística en función de estas. Dado que tenemos las máscaras de piel ideales podemos computar dos histogramas (Uno de piel y uno de fondo) para luego comparar los píxeles de entrada. Para ello primero será necesario encontrar una base de color donde se puedan representar de la mejor forma los colores en tan solo dos dimensiones. Más adelante se comentan los métodos utilizados.

Finalmente, si se normalizan estos histogramas según el número de píxeles de piel o fondo, tendremos las probabilidades de croma condicionadas. Y siguiendo el criterio de decisión bayesania

$$P(C|p) * P(p) \underset{H_1}{\overset{H_0}{\gtrless}} P(C|f) * P(f) \tag{2}$$

De forma equivalente a (2), y más eficiente en términos de computación

$$\log(P(C|p) * P(p)) - \log(P(C|f) * P(f)) \underset{H_1}{\overset{H_0}{\gtrless}} 0 \tag{3}$$

Como las probabilidades de piel y de fondo serán siempre las mismas podemos eliminarlas y simplemente adjustar el umbral. Con lo que nuestra decisión final sería

$$\log(P(C|p)) - \log(P(C|f)) \underset{H_1}{\overset{H_0}{\gtrless}} \alpha_{opt} \tag{4}$$

En base a esto, finalmente calcularemos el *Recall* y *Precision* para dar con el *F-Score* y tener un sistema de medida de rendimiento de nuestro programa.

$$Precision = \frac{P}{TP} \tag{5}$$

$$Recall = \frac{P}{T} \tag{6}$$

$$F_{Score} = 2 * \frac{Precision * recall}{Precision + Recall} \tag{7}$$

El documento se divide en (II) Obtención de Imágenes de Entrenamiento, (III) Preprocesado, (IV) Obtención de Histogramas y Normalización, (V) Decisión, (VI) Postprocesado, (VII) Resultados, (VIII) Análisis de Resultados y (IX) Posibles mejoras del sistema

II. Obtención Imágenes de Entrenamiento

En este priner paso, lo primero que se debe hacer es obtener las imágenes de prueba junto con sus respectivas máscaras. Para ello, lo primero es definir los vectores que guardarán esta información

```
1
2   %Nos situamos en el directorio de las imagenes
3   cd ('.\Images');
4   list_images=dir('*.jpg');
5
6   %Definicion de vectores de longitud list_images
7
8   images = cell(1, length(list_images));
9   images_b = cell(1, length(list_images));
10  mask_file = cell (1,length(list_images));
```

Seguidamente, guardar las imágenes en nuestro vector. El coste computacional de la iteración por todo el vector es *O(n)*, siendo n el número de imágenes.

```
1  for i = 1 : length(list_images)
2      image_file = imread(list_images(i).name);
3      images{i}=image_file;
4  end
```

Se haría lo mismo con las máscaras (accediendo al respectivo directorio y cogiendo los archivos .bmp) y se tendrían los dos vectores *images* y *mask-file* ya inicializados. Toda esta lógica está implementada en el fichero 'algo1'.

III. Preprocesado

Para poder abordar el problema por medio de un algoritmo, el primer desafío que hemos encontrado ha sido saber cómo íbamos a tratar las imágenes y preparar todo el escenario para poder desarrollar de manera eficiente y efectiva la detección de píxeles de piel.

Lo primero que hemos hecho ha sido cargar y guardar los elementos necesarios para poder establecer el modelo (ya se ha explicado de manera detallada este proceso). Estos son, las imágenes con las que desarrollar los modelos de histogramas y las máscaras ideales (que separan piel y fondo) de las imágenes.

Posteriormente, se han aplicado estas máscaras ideales a las imágenes originales para tener, por separado, dos conjuntos de nuevas imágenes; unas que solo tendrían pixeles de la piel de las imágenes originales, y otras que solo tendrán pixeles del fondo. Esta separación se realiza para, de esta forma, poder trabajar por separado con piel y fondo y crear dos histogramas diferentes para las dos situaciones. Esto nos ayudará, en última instancia, a desarrollar el criterio de decisión que separará pixeles de piel y de fondo.

```
for i = 1 : length(list_images)

    masks_file = imread(list_masks(i).name);
    mask_file{i}=int8(masks_file);

    images_b{i}(:,:,1) = images{i}(:,:,1).*uint8(
        masks_file);
    images_b{i}(:,:,2) = images{i}(:,:,2).*uint8(
        masks_file);
    images_b{i}(:,:,3) = images{i}(:,:,3).*uint8(
        masks_file);

    images{i}(:,:,1) = images{i}(:,:,1).*uint8(1-
        masks_file);
    images{i}(:,:,2) = images{i}(:,:,2).*uint8(1-
        masks_file);
    images{i}(:,:,3) = images{i}(:,:,3).*uint8(1-
        masks_file);
```

Una vez tenemos todo esto, otro desafío que se nos presentó fue saber qué espacio de representación de imagen utilizaríamos para crear y estudiar los dos histogramas. Teníamos varias opciones (RGB, YCrCv, HSV, Lab, etc.), pero como el objetivo era extraer las dos componentes de crominancia de las imágenes para luego hacer el histograma, hemos escogido el espacio YCrCv. Este espacio nos permite visualizar las dos componentes de croma de una imagen de manera muy explicita. Gracias a esto, se puede observar que los pixeles de piel y los de fondo se aglutinan en regiones diferentes de este espacio, ya que tienen características de croma muy dispares. De esta manera, este espacio de color resulta un método atractivo para la modelización del color de la piel.

Por tanto, aplicamos una transformación en los dos conjuntos de imágenes, para pasar del espacio RGB a el espacio YCrCv. Obviamos la componente de luminancia y guardamos las componentes de crominancia de todas las imágenes.

```
images{i} = rgb2ycbcr(images{i});
images_b{i} = rgb2ycbcr(images_b{i});

%% Componentes Cb y Cr de las imagenes %%
cb = images{i}(:,:,2);
cr = images{i}(:,:,3);

cbb = images_b{i}(:,:,2);
crb = images_b{i}(:,:,3);
```

Una vez tenemos guardadas estas componentes ya estamos en condiciones de empezar a desarrollar los histogramas y se da por terminado el preprocesado.

IV. Obtención histogramas y normalización

Para poder establecer una distancia entre el color de pixel entrante y su respectiva probabilidad primero es necesario crear el modelo estadístico que permita crear esta referencia de comparación. Es por ello que se crean dos histogramas bidimensionales, uno de los píxeles de piel y otro de los de fondo. Los dos ejes serán los componentes de crominacia (Cr y Cb) y cada valor tendrá asociado una cantidad de veces que aparece. Por lo que, al normalizarlos por los respectivos números de píxel tendríamos $P(C|p)$ y $P(C|f)$.
Se podría haber decidido simplemente tener un histograma de los píxeles de piel, pero lo consideramos así para poder establecer el criterio de decision en (4).

Para poder crear el histogrma se utiliza la función *histcounts2*. Es necesario pasarle dos vectores denominados Edges para poder indicar como mapear los valores de crominancia a la base del histograma. $N(i,j)$ cuenta la variable $[Cr(k), Cb(k)]$ si $X_{edges}(i) \leq Cr(k) \leq X_{edges}(i+1)$ y si $Y_{edges}(i) \leq Cb(k) \leq Y_{edges}(i+1)$.

En nuestro caso, al definir ambos Edges como un vector equiespaciado (por 1) de 1:257 el mapeo que se realizará será directo. Si $[Cr(i), Cb(j)] = [167, 245]$, el valor se almacenará en la posición (167,245) del histograma. Contemplando que los valores de Cr y Cb van de 16 a 240 el valor óptimo de bins es 256 (Viene ímplicito en la declaración de ambos vectores).

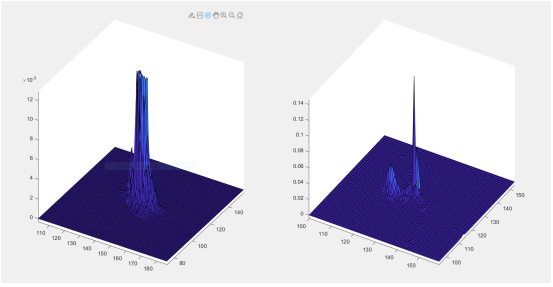

Figura 1. Histogramas normalizados.

Otro detalle a tener en cuenta a la hora de realizar los historamas es que aparecerán muchos píxeles negros tras haber aplicado la máscara. Estos píxeles no deberían tenerse en cuenta a la hora de crear los histogramas, por lo que existen distintas formas de plantearlo. Se podría hacer que el histograma no tuviera en cuenta los píxeles negros y solo los píxeles de interés. Otra opción, y la utilizada, es eliminar el valor del histograma para este color en cada iteración. Se escogió esta opción para poder entender mejor cómo funcionaba el flujo de creación de los histogramas y para poder modificar los valores. Eliminar los máximos del histograma general no es viable, ya que se eliminarán valores que no tienen porque ser el negro (Sobretodo en iteraciones avanzadas) por lo que se debe eliminar el máximo de cada histograma guardándolo en una variable auxiliar para modificarlo. Como es posible que existan píxeles de piel de color negro (en el fondo igual) debemos restar solamente el número de unos que tiene la máscara a este máximo; No ponerlo a 0.

```
1
2 %% Vectores para crear los histogramas basandose en
      el numero de bins(Edges)
3 xv = 1:bins+1;
4 yv = 1:bins+1;
5
6 %%Primera iteracion -> No es necesario sumar ni
      almacenar en variable auxiliar
7 if(i == 1)
8     histcount = histcounts2(cb,cr,xv,yv);
9     %% Eliminamos del maximo los valores negros que
          resultan de haber aplicado las mascaras
          ideales a las imagenes %%
10    histcount(histcount==max(histcount(:))) = max(
          histcount(:))-sum(mask_file{i}(:));
11
12
13    histcount_b = histcounts2(cbb,crb,xv,xv);
14    histcount_b(histcount_b==max(histcount_b(:))) =
          max(histcount_b(:))-sum(1-mask_file{i}(:))
          ;
15 else
16    [histcount_aux, Xedges,Yedges, binX, binY] =
          histcounts2(cb,cr,xv,xv);
17    histcount_aux(histcount_aux==max(histcount_aux
          (:))) =max(histcount_aux(:))-sum(mask_file{
          i}(:));
18
19    %% Sumamos al histograma total el histograma de
          cada imagen %%
20    histcount = histcount + histcount_aux;
21
22    histcount_aux = histcounts2(cbb,crb,xv,xv);
23    histcount_aux(histcount_aux==max(histcount_aux
          (:))) = max(histcount_aux(:))- sum(1-
          mask_file{i}(:));
24    histcount_b = histcount_b + histcount_aux;
25 end
26
27
28 %% Normalizacion de los histogramas por el numero
      de pixeles de piel y fondo respectivamente %%
29 histCountPiel = histcount/total_pixels_piel;
30 histCountFondo = histcount_b/total_pixels_fondo;
```

V. Decisión

Una vez tenemos los histogramas normalizados de piel y fondo estamos en condiciones de proponer un esquema de decisión para poder clasificar pixeles de imágenes como piel o no piel. Como podemos observar en los resultados de los dos tipos de histograma, estos tienen sus valores máximos en diferentes niveles de crominancia. También cabe destacar que, al crear estos histogramas normalizados a partir de una secuencia de entrenamiento, estamos creando unos modelos de probabilidad discreta de piel y fondo en el espacio de crominancia.

$$(P(c)_{piel} = \frac{Hist_{piel}(c)}{N_{pixelsPiel}} \tag{8}$$

donde c es el vector de los valores de crominancia.
Pasaría lo mismo con la estimación de probabilidad de fondo:

$$(P(c)_{fondo} = \frac{Hist_{fondo}(c)}{N_{pixelsFondo}} \tag{9}$$

Por este motivo, inicialmente se planteó usar el método de decisión y clasificación Bayesiano. Las probabilidades $P(c)_{piel}, P(c)_{fondo}$ son probabilidades condicionadas $P(c|piel), P(c|fondo)$ ya que hemos construido los histogramas a partir de los respectivos pixeles de piel y de fondo de las imágenes de entrenamiento (que hemos separado en piel y fondo). Por tanto, estas probabilidades serían la probabilidad de observar el vector c sabiendo que el pixel es piel o fondo. En nuestro algoritmo y criterio de decisión queremos saber $P(piel|c), P(fondo|c)$. Ya que las muestras serán imágenes, podremos saber el vector c de cada pixel de las imágenes de datos, y con estas probabilidades clasificar por medio de Bayes los pixeles.

Para obtener las probabilidades condicionadas al vector de las muestras se puede utilizar la regla de Bayes:

$$P(piel|c) = \frac{P(c|piel) * P(piel)}{P(c|piel) * P(piel) + P(c|fondo) * P(fondo)} \tag{10}$$

Ídem para el fondo.

Por tanto, necesitamos, en un principio, hallar varias probabilidades; todas ellas se pueden encontrar de los histogramas normalizados y de los datos de las imágenes de entrenamiento. Como realmente nuestro objetivo principal es montar un clasificador y detector de Bayes, necesitamos solamente encontrar de forma explícita $P(c|piel)$ y $P(c|fondo)$. Esto es porque apenas estamos interesados en la relación:

$$\frac{P(piel|c)}{P(fondo|c)} = \frac{P(c|piel) * P(piel)}{P(c|fondo) * P(fondo)} \tag{11}$$

Ya estamos en condiciones de establecer el detector, porque estas probabilidades condicionadas a piel y fondo ya la tenemos gracias a la creación de los histogramas normalizados desarrollados a partir del entrenamiento. Montando finalmente el detector Bayesiano tendremos:

$$\log(P(c|piel)) - \log(P(c|fondo)) \underset{H_1}{\overset{H_0}{\gtrless}} \alpha_{opt} \quad (12)$$

Como se mencionaba en la introducción está decisión se pasa a escala logarítimica para mejor rendimiento computacional (Sumas y no multiplicaciones) y se acaba obteniendo (4).

Para aplicar todo este proceso en nuestro a algoritmo, hacemos el histograma de la imagen de entrada e inicializamos la máscara que separará los píxeles de piel de los píxeles de fondo. Seguidamente, analizaremos todos los píxeles de la imagen y los clasificaremos en piel o no piel. Para ello, miramos el nivel de crominancia del píxel de entrada (que valores de bin tiene en el histograma de la imagen cada valor de croma). Después para estos bins concretos hallamos $P(c|piel)$ y $P(c|fondo)$ de los histogramas normalizados que ya hemos creado con anterioridad y aplicamos nuestro criterio de decisión. Si se decide piel se pone en esa posición de píxel (i, j) un "1" en la máscara de esa imagen.

```
1
2 imagen_prueba = rgb2ycbcr(im);
3 cb_im = double(imagen_prueba(:,:,2));
4 cr_im = double(imagen_prueba(:,:,3));
5 [N,Xedges,Yedges,binX,binY] = histcounts2(cb_im,
      cr_im,xv,yv); %% Se crea el histograma de la
      imagen, usando los Edges determinados %%
6 [li, wi, di] = size(imagen_prueba);
7 mask_image=zeros(li, wi);  %% Se inicializa la
      mascara, a crear, de la imagen %%
8
9 %% numero de pixeles del radio de los elementos
      estructurantes usados para el post-procesado %%
10 r_o =4;
11 r_a = 5;
12
13 %% Bucle principal de decision; si un pixel de la
      imagen es piel o fondo %%
14 for i = 1 : wi
15     for j = 1 : li
16         pos_x = binX(j,i); %% Localizamos cada
               pixel de la imagen %%
17         pos_y = binY(j,i);
18
19         %% Detectamos el valor de cada histograma(
               puede ser interpetado como una
               probabilidad) para el valor Cb Cr de
               ese pixel %%
20         prob = histcountPielNormalized (pos_x,
               pos_y);
21         prob_b = histcountFondoNormalized(pos_x,
               pos_y);
22
23          %% Si se cumple que la razon entre la
                probabilidad de piel de ese nivel y la
                probabilidad de fondo supera cierto
                treshold decidiremos piel %%
24         if(log(prob)-log(prob_b)>= treshold  )
25             mask_image(j,i) = 1;  %% Actualizamos
                   mascara para decidir piel en el
                   pixel correspondiente %%
26         end
27     end
28 end
```

VI. Postprocesasdo

De cara a mejorar el criterio de F-Score se han aplicado los operadores morfológicos de cierre y apertura de forma consecutiva a la máscara obtenida. Es importante entender en qué consisten estos operadores y cómo afectan al procesado. Se definen los operadores según el orden: Apertura y Cierre.

$$A \circ B = (A \ominus B) \oplus B \quad (13)$$

$$A \bullet B = (A \oplus B) \ominus B \quad (14)$$

Por lo que se ve en (8) y (9) en ambos operadores se aplican dilataciones y cierres pero se cambia el orden. En una apertura, por tanto, al aplicar antes una erosión se eliminarán los elementos de blanco menores al tamaño del elemento estructurante. En el cierre, lo mismo pero para los negros.
Por tanto, la apertura nos será muy útil de cara a borrar los elementos del fondo que no se han acabado de detectar como fondo. Por contraparte, el cierre será útil para los píxeles de piel que no se han detectado como total. A la hora de escoger el tamaño del elemento estructurante, debe tenerse en cuenta que la apertura debe eliminar el máximo de elementos del fondo pero sin llegar a eliminar segmentos estrechos de piel (Puede causar errores en los dedos si estos son muy finos). En el caso del cierre, se debe intentar que no se lleguen a solapar segmentos de piel. De nuevo, los dedos son muy susceptibles a esto si no están muy separados.

Tras probar e intentar optimizar con distintas formas y tamaños, el mejor F-Score obtenido se dio con elementos estructurantes con forma de disco y de un radio de 4 para el cierre y 5 para la apertura (En píxeles).

```
1
2 SE = strel('disk',r_o);
3 SE2 = strel ('disk', r_a);
4 mask_image = imclose(mask_image, SE);
5 mask_image = imopen(mask_image, SE2);
```

Finalmente, notar que tras este preprocesado el F-Score se aumentó en 2,034. Un incremento muy a tener en cuenta. Sobretodo, la apertura es de gran interés puesto que no se detectaba demasiado bien partes del fondo.

VII. Resultados y Análisis

En esta sección se describen los diferentes experimentos realizados al prototipo, junto a los resultados obtenidos representados mediante gráficas y tablas. Nuestro objetivo es analizar los resultados en base a dos conceptos. El primero es maximizar el F-Score y el otro el tiempo de ejecución.

VII-A. Maximización del F-Score

Para poder maximizar el F-Score (7) es necesario optimizar el Treshold a mencionado en (4), junto con los elementos estructurantes mencionados en el apartado de Post-Procesado.

Con este fin, aprovechando que a la función 'algo2' se le pasa como parámetro el correspondiente umbral, podemos establecer una estructura de bucle en el programa prinicpal alterando el valor de este en cada iteración. El código relacionado con esta lógica es el siguiente.

```
%%Creacion de histogramas
[histCountPiel, histCountFondo,xv,yv] = algo1();
cd('..');

treshold = -3; %%Se establece el Treshold inicial

%%Vectores para representar graficas
x = 1:60;
y = 1:60;
y_p = 1:60;
y_r = 1:60;
i = 0;

%%Iteraciones con diferente valor de Treshold
while(treshold<=3)
    i=i+1;

    algo3(treshold,xv, yv, histCountPiel,
        histCountFondo); %%Obtencion mascaras con
        correspondiente Treshold

    [f_score, precision, recall] = algo4(); %%
        Obtencion de F-score, precision y recall
    cd('..');

    %%Asignacion de valores en los vectores para la
         representacion
    x(i) = treshold;
    y(i) = f_score;
    y_p(i) = precision;
    y_r(i) = recall;
    treshold= treshold + 0.1;
end

%%Plot de las grAficas
subplot(2,2,1);
plot(x, y);
title('F-Score');

subplot(2,2,3);
plot(x, y_r);
title('Recall');

subplot(2,2,4);
plot(x, y_p);
title('Precision');
```

En el código mostrado se itera desde -3 a 3 con un incremento de 0.1 pero se ha ido afinando hasta modificar números del orden de 10^{-3} . Finalmente, tras diversas pruebas e iteraciones se llegó a la conclusión que el valor óptimo para el umbral sería $a_{opt} = 0{,}0549$, teniendo $Fscore(a_{opt}) = 87{,}263$

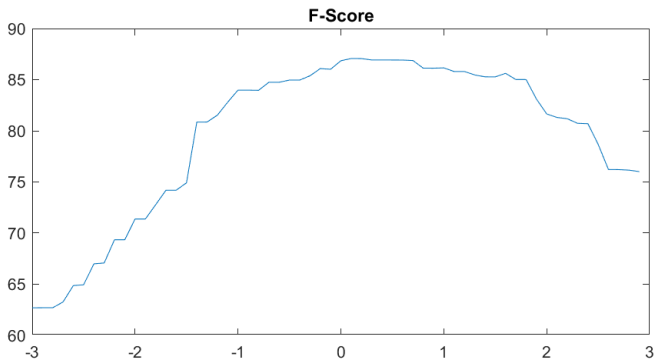

Figura 2. F-Score en función del Treshold.

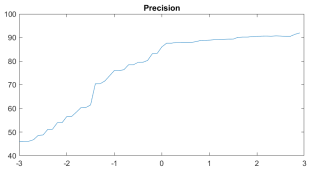

Figura 3. Precisión en función del Treshold

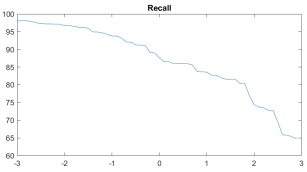

Figura 4. Recall en función del Treshold

Como se observa en las gráficas, los valores donde el F-Score es mayor caen en las zonas donde $a \in [0, \frac{1}{2}]$.
Si se compara con las gráficas de Precisión y Recall (Que son inversamente proporcionales) vemos que en esta misma zona los valores, a pesar de no ser los más altos de su entorno, si son los más altos teniendo en cuenta ambos parámetros. Que, al final, es de lo que se trata el F-Score. Con valores de Umbral muy pequeños la precisión es muy pequeña dado que la cantidad de elementos considerados piel es muy grande. Por lo que, evidentemente el Recall será elevado. Ocurre lo contrario para umbrales mayores.

Para poder encontrar las estructuras morfológicas adecuadas se hizo de la misma forma. Con un bucle de prueba en la función 'algo2'. Ya se han detallado los elementos en el apartado VI.

VII-B. Tiempo de ejecución

Con la intención de tener el menor tiempo de ejecución posible para nuestro procesado de imágenes, hemos tenido mucho cuidado con el tipo de operaciones que realizamos en nuestro algoritmo y el numero de estas operaciones.

Primero analizamos el tiempo que tarda en ejecutarse el algoritmo de decisión, que crea las máscaras para cada imagen, para todas las imágenes de validación. Debemos mencionar que el coste computacional de nuestro sistema para cada imagen será $\max(T(algo2()), T(imwrite())$.
Teniendo en cuenta que
T(algo2()) = $\max(O(nm), T(histocounts2()), T(rgb2crcb()))$.
Dado que el sistema itera por todos los píxeles (n.m) y fuera del bucle llama a las funciones 'histcounts2' y 'rgb2crb'. No se ha conseguido información del coste de estas funciones, por lo que entendemos que

$$T_{Imagen} = T(algo2()) = O(nm); \tag{15}$$

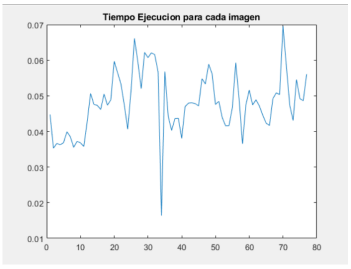

Figura 5. Tiempos de ejecución para cada imagen.

La media aproximada del tiempo de ejecución por imagen nos da: $T_{medio} = 0{,}0488s$.

Analizamos ahora el tiempo que se tarda en ejecutar el código que genera el modelo de histogramas de piel y fondo a partir de la secuencia de entrenamiento. La complejidad computacional asintótica del sistema sería $O(k * f)$, siendo k el número de imágenes de entrenamiento y f la función con computación más compleja de las usadas (imread, histcount2, rgb2crcb). Tras realizar 12 medidas, la media del tiempo es: $T_{modelo} = 4{,}18s$.

También, partiendo de (15), el tiempo en ejecutar la creación de todas las máscaras de las imágenes de validación será $O(inm)$, siendo i el número de imágenes de validación y n y m los respectivos píxeles. $T_{procesado} = 5{,}96s$.

Por ultimo el tiempo total de procesar las imágenes de validación, las mascaras y calcular el Fscore es: $T_{total} = 9{,}208s$.

Estos tiempos estimados se han establecido haciendo la media entre 15 valores del mismo tiempo de proceso. Mencionar también que para calcular estos tiempos de ejecución se ha utilizado un PC con procesador Intel(R) Core(TM) i7-8550U CPU @ 1.80GHz, 1992 Mhz, 4 procesadores principales, 8 procesadores lógicos. Con 16GB de memoria RAM y corriendo Windows 11. También, cinfigurado parra no acceder a instrucciones ni direciones guardadas en caché.

VIII. Posibles Mejoras del Sistema

Varias mejoras podrían ser implementadas para mejorar el funcionamiento del algoritmo y el F-sccore. Sobre todo en lo que respecta a la decisión del espacio de representación y en la forma de caracterizar la distribución probabilidad de los pixeles de piel.

Una posible mejora que podríamos haber implementado y probado es utilizar otro tipo de modelo de representación de color de las imágenes. Para poder detectar colores y regiones a través de histogramas resulta que utilizar el modelo HSV funciona bastante mejor. Las siglas H, S y V corresponden a Tono (hue), Saturación (saturation) y valor (value) respectivamente. Este tipo de modelo resulta muy conveniente para detectar tipos de colores parecidos y específicos, ya que colores parecidos donde solo cambiase la saturación o si tiene más o menos sombra, igualmente se mantendrían con valores HSV parecidos. Ocuparían una región en el modelo hexagonal HSV muy concreta y muy diferente a otro tipo de colores que podrían pertenecer al fondo, lo que beneficiaria mucho en la detección de piel. Sobre todo, con imágenes reales, donde los colores de piel pueden estar muy influenciados por saturaciones, sombras u otros elementos cercanos con colores similares, este tipo de representación hubiese sido muy beneficiosa a la hora de hacer una clasificación por medio de histogramas.

Hablar de métodos de detección y clasificación sin hablar de métodos paramétricos es incompatible. Este tipo de métodos generan modelos de piel mas compactos que permiten generalizar e interpolar los datos de entrenamiento. Con este tipo de modelos aproximaríamos de manera mas precisa y a nivel más general la distribución de probabilidad de los pixeles de piel. Destacaremos dos modelos: el modelo simple de Gauss y el modelo de mezcla de Gaussianas. Con el modelo de Gauss aproximaríamos la distribución de color de piel por una función de densidad de probabilidad de Gauss. La PDF estaría definida como

$$p(c|piel) \sim N(\mu_s, \mathcal{E}) \tag{16}$$

Donde c es el vector de color, μ_s es el vector de medias y $\mathcal{E}$ es la matriz de covarianza. Estos dos últimos parámetros se estimarían a partir de la secuencia de entrenamiento por medio de la maximización de la función Likelihood.

Otra opción seria usar un modelo GMM para estimar la densidad de probabilidad de la piel. Este modelo es más sofisticado y permite estimar distribuciones mas complejas. Es una generalización del modelo descrito con anterioridad. En este caso la función de densidad se aproximaría por:

$$p(c|piel) = \sum_{i=1}^{k} p_i(c|piel) \tag{17}$$

Donde k es el numero de gaussianas que se van a utilizar, es la ponderación que recibe cada Gaussiana en la mezcla y son las funciones de densidad de probabilidad de gauss. Para estimar todos los parámetros a partir de las muestras de entrenamiento se podría usar el algoritmo EM(Expectation-Maximization), que supone el conocimiento de k (número de gaussianas). Por tanto, antes de aplicarlo seria necesario un estudio sobre el k óptimo en nuestro caso.

Finalmente, comentar que se podría establecer un modelo basado en regiones. Este método se intentó aplicar pero tal y como lo planteamos no dio resultados mejores a los previamente obtenidos. La lógica de esta ampliación sería obtener los valores del histograma cercanos al del píxel entrante y hacer una media de los valores dados y compararlo un umbral.