# PIV
# Informe Pràctica de Programació

CSD: Color Structure Descriptor

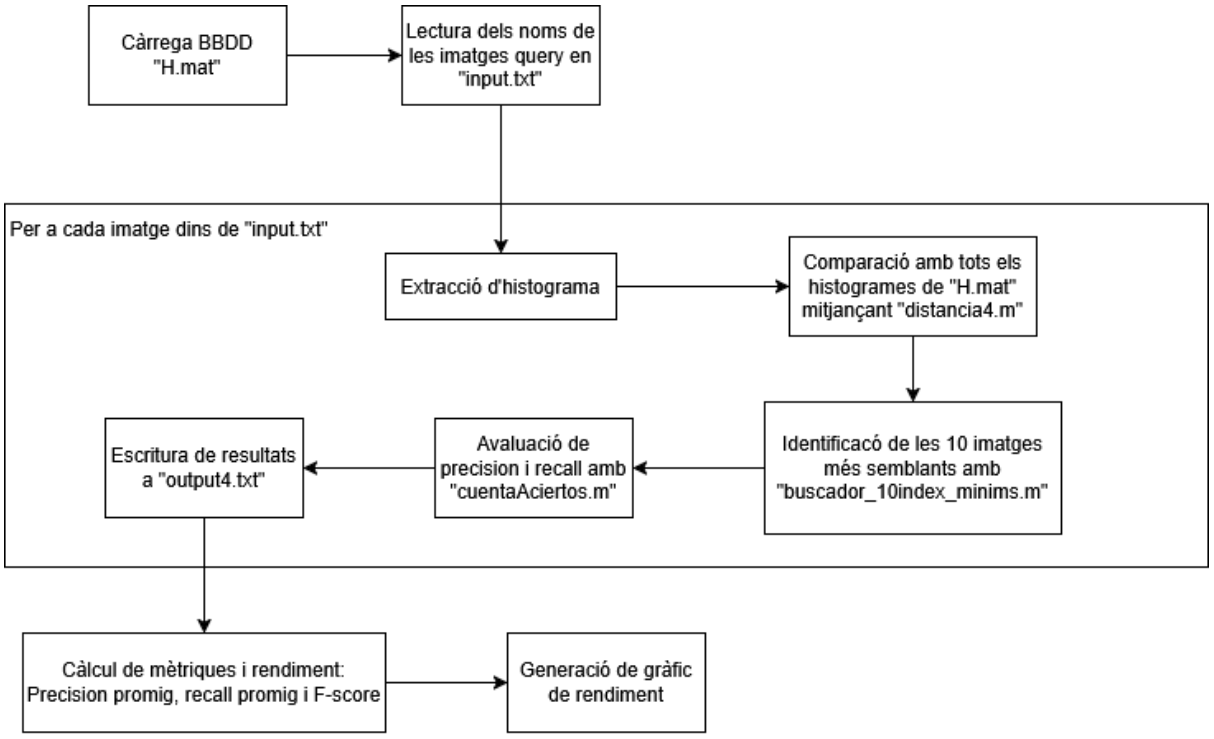
Curs 2024-2025
Quadrimestre de primavera
Maig de 2025

# Índex

# 1. Descripció global del sistema

Aquesta pràctica es una continuació directa del sistema desenvolupat en la anterior pràctica; en la que es va desenvolupar un CBIR (del anglès “Content-Based Image Retrieval”).

En aquell primer intent, l’algorisme caracteritzava les imatges únicament mitjançant el seu histograma de nivells de grisos, sense tenir en compte la seva informació de color. Estava composat pel següent diagrama de blocs:



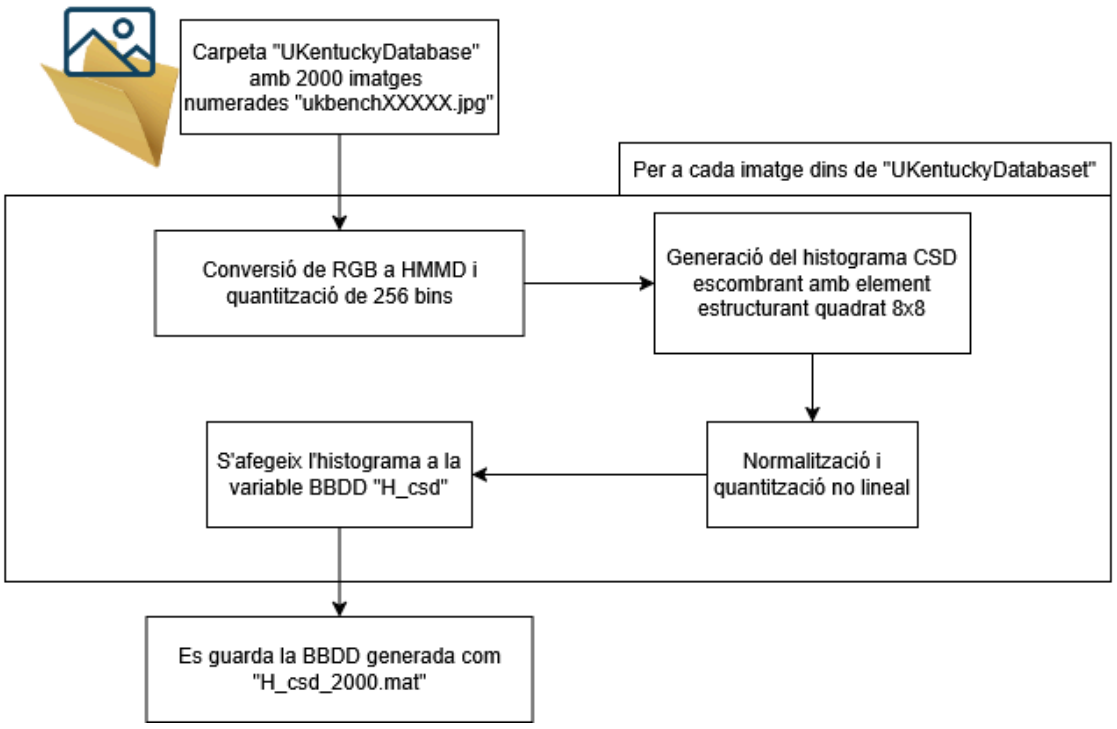

Aquesta segona pràctica se centra en com obtenir una millor base de dades per millorar la cerca d’imatges. La nova base de dades serà d’histogrames generats a partir del descriptor de color CSD (del anglès “Color Structure Descriptor”); és a dir, histogrames que tenen en consideració, com a mínim, el color que presenten les imatges. El treball principal de la segona pràctica ha sigut el analitzar, entendre i implementar aquest descriptor per generar una nova base de dades de histogrames. S’ha aprofitat el sistema desenvolupat en la primera pràctica per avaluar els resultats.

El CSD descriu una imatge tenint en compte tant la distribució global de colors com la seva estructura espaial local. A diferència dels histogrames de color “normals”, que només compten quants cops apareix cada color, el CSD també té en compte on apareixen.

Funciona escanejant la imatge amb una finestra, o bola, (element estructurant) de 8×8 píxels, i incrementa un comptador per a cada color que està present (no la quantitat) dins d’aquesta finestra. Abans de fer l’escaneig també es pot fer un delmat de la imatge per estalviar cost computacional. El resultat és similar a un histograma però amb sensibilitat a la textura i la dispersió dels colors.

Es calcula sobre l'espai de color HMMD quantificat no uniformement, i es pot generar en mides de 256, 128, 64 o 32 bins. La quantificació final dels valors de cada bin es fa de manera no lineal per afavorir la discriminació en valors petits.

Tenint tot això en consideració, el primer sistema desenvolupat queda descrit en el següent diagrama de blocs i fases:




Dins del bucle principal a `Prog2_CSD_Feature_Extraction.m`, per a cada imatge es fa:

1. **Conversió i quantització RGB a HMMD**

Es passa de RGB a HMMD mitjançant la funció **`prog2_rgb2hmmd`** i seguidament es quantitzada, amb 256 bins, tal i com indica l'estàndard mitjançant la funció **`prog2_hmmd_quantize`**.

La primera funció s'ha programat de forma que es pugui escollir que les components de la imatge de sortida siguin (H, D, S) (Hue, Difference i Sum) o bé (H, M, M) (Hue, Min, Max). S'ha treballat utilitzant el mode "HDS".

2. **Escombrat de la imatge**

Després "s'escombra" la imatge amb un element estructurant 8x8. Cada posición "k" del histograma CSD representa el nombre de boles 8x8 en las que apareix al menys una vegada el bin "k". No compta quantes vegades apareix cada color en total. Només si apareix o no.

3. **Normalització i quantització no lineal**

Es normalitza l'histograma CSD generat i es quantitza de forma no lineal tal i com indica l'estàndard.

4. **Emmagatzemament del histograma**

Es guarda l'histograma dins de la variable "H_csd".

Finalment, es guarda la base de dades en el fitxer "H_csd_2000.mat"

Com es pot apreciar, en aquest primer intent d'implementar el CSD **no** s'ha fet la part opcional d'aplicar delmat. Sí que s'ha aplicat en una de les versions alternatives que es va desenvolupar més endavant. Queda explicat a l'apartat 3 d'aquest document.

Per veure el detall del bucle principal i les funcions que intervenen en el procés d'extracció de característiques consulti l'annex A1.1.

Per acabar, un cop amb la base de dades generada, se li aplica el sistema de classificació desenvolupat a la pràctica 1, definit en el fitxer `Prog2.m`.

L'únic canvi destacable respecte la última entrega es que s'han provat diverses distàncies, implementades a `distancia4_variable.m`. De forma experimental s'ha decidit treballar amb la distància Hellinger donat que ha sigut la que ha donat millor F-score. I es amb aquesta distància que s'ha treballat per a totes les millores posteriors.

La resta de funcions no han sigut modificades, però també s'adjunten al final d'aquest treball, a l'annex A1.2.

Abans de analitzar resultats i per comoditat a l'hora d'explicar les versions alternatives a l'apartat 3 només indicar que a aquesta configuració d'extracció de característiques se li ha denominat "256_base".

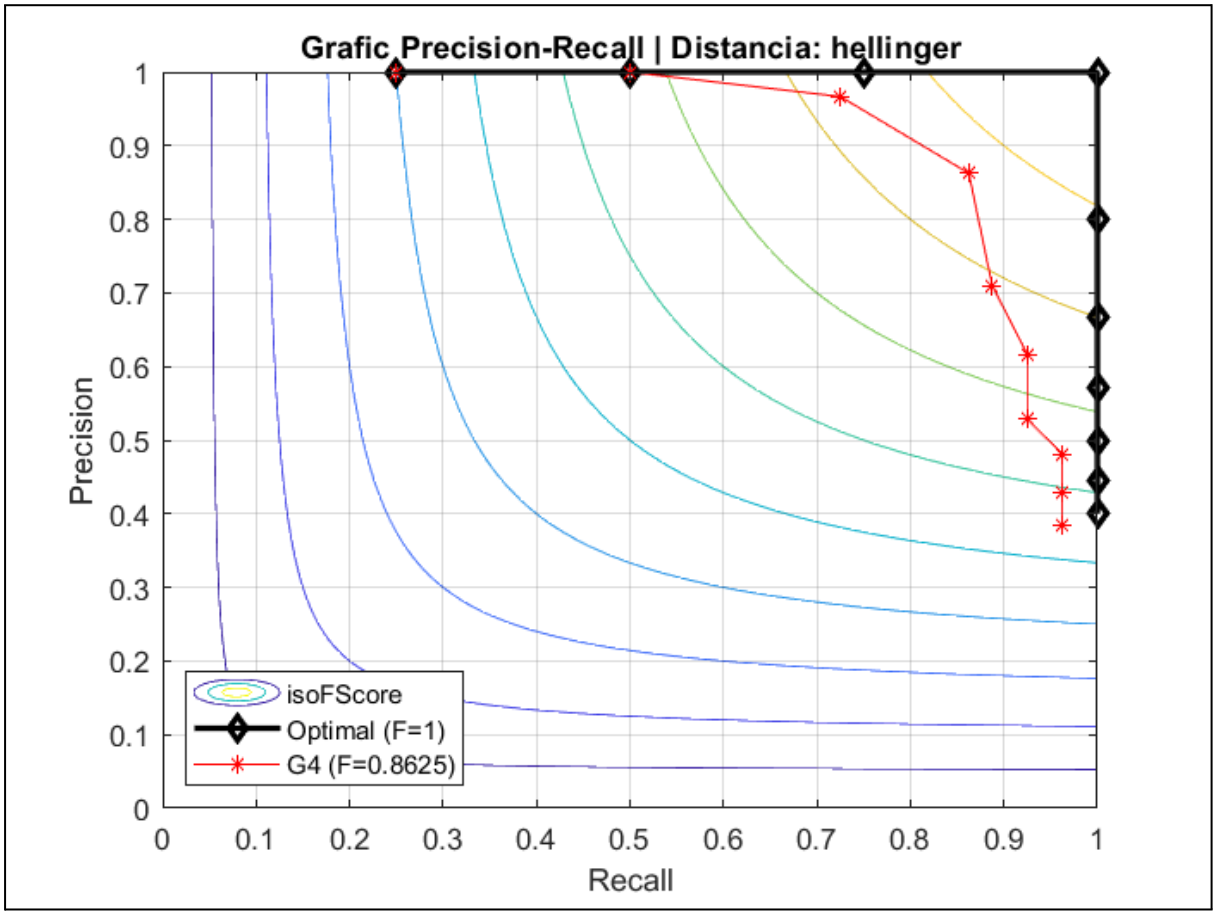
Aquesta configuració es diferencia de les altres perquè:

- No ha estat optimitzada
- No aplica delmat abans de l'escombrat
- S'ha utilitzat un element estructurant de tamany 8x8
- S'ha treballat amb histogrames de 256 bins

És amb aquesta configuració la que es va generar la primera bases de dades, de la qual va sorgir el fitxer `output4.txt` que es va entregar l'última sessió de laboratori dedicada a aquest projecte.

# 2. Resultats del sistema

Amb la configuració “256_base” i distància Hellinger s’ha obtingut la següent corba “Precision-Recall”:




**S’ha aconseguit un F-Score de 0,8625**

A nivell de cost computacional, aquesta configuració va trigar aproximadament unes 2 hores i 50 minuts en generar la base de dades “H_csd_2000.mat”.

```
>> Prog2_CSD_Feature_Extraction
Hora inicio procesado 2000 imagenes: 21/04/25-12:41
Elapsed time is 10245.003427 seconds.
Hora finalizacion procesado 2000 imagenes: 21/04/25-15:32
>>
```

La base de dades es una matriu de 2000 files per 256 columnes i ocupa uns 84 kiloBytes.

# 3. Anàlisi de resultats i conclusions

Una F-Score de 0,8625 es una millora sustancial respecte la F-Score de 0,5 obtinguda en la primera part d'aquest projecte.

Aquest F-score, del 86%, ens indica que tant el precision com el recall son relativament alts; és a dir, que el sistema encerta la majoria de vegades i que no deixa passar moltes imatges rellevants a l'hora de fer una query.

Es un resultat esperat. Al utilitzar informació de color a l'hora de construir els histogrames ja esperàvem que el resultat final millorés bastant. No esperàvem que millorés poc, com per exemple una F-Score de 0,6. Tampoc esperavem que tingués una F-Score de 0,95 o més ja que llavors aquest descriptor seria el que tothom utilitzaria pels bons resultats que dona i la resta no tindrien raó d'existir.

## Versions alternatives

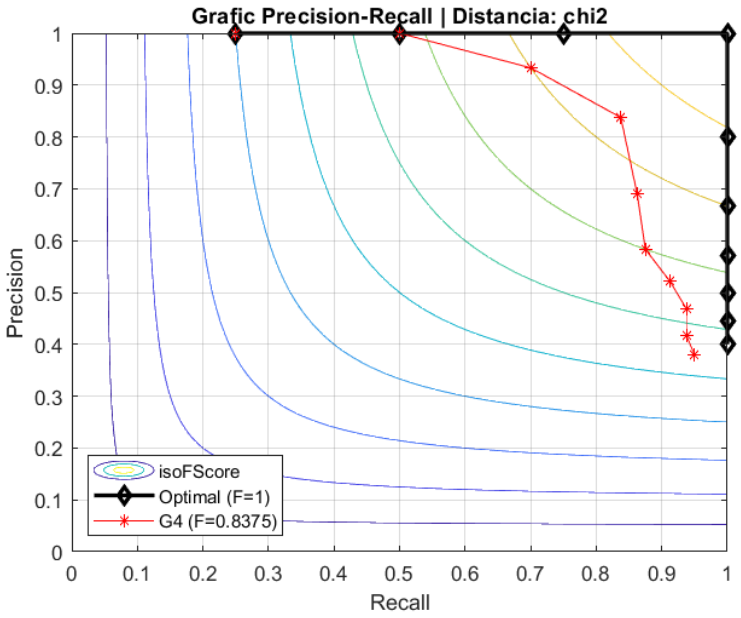
### Configuració "256_base" amb diferents distàncies

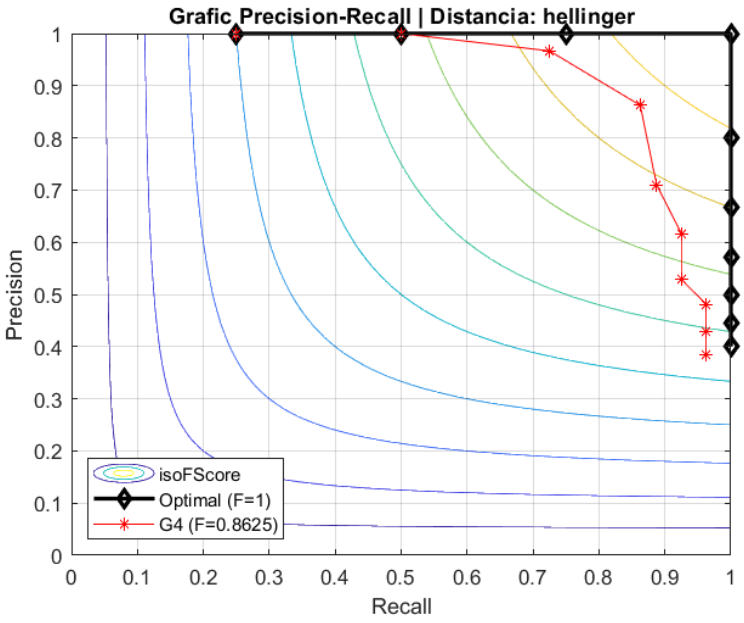
Aquesta configuració ha sigut la primera en ser desenvolupada i es la que ens va generar la primera base de dades "H_csd_2000.mat". A l'hora d'avaluar-la hem tingut en consideració des d'un inici el provar diferents definicions de distància donat que era quelcom que va quedar pendent a la primera part del projecte.

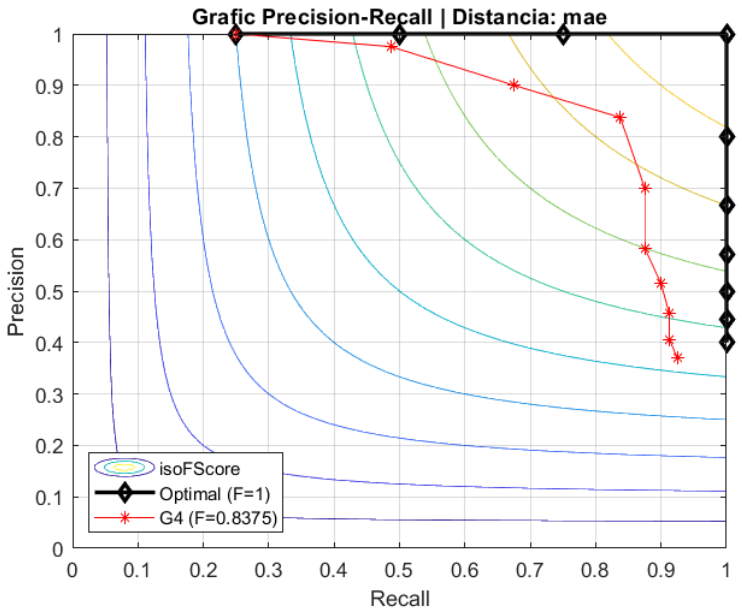
A `distancia4_variable.m` s'han implementat les distàncies:

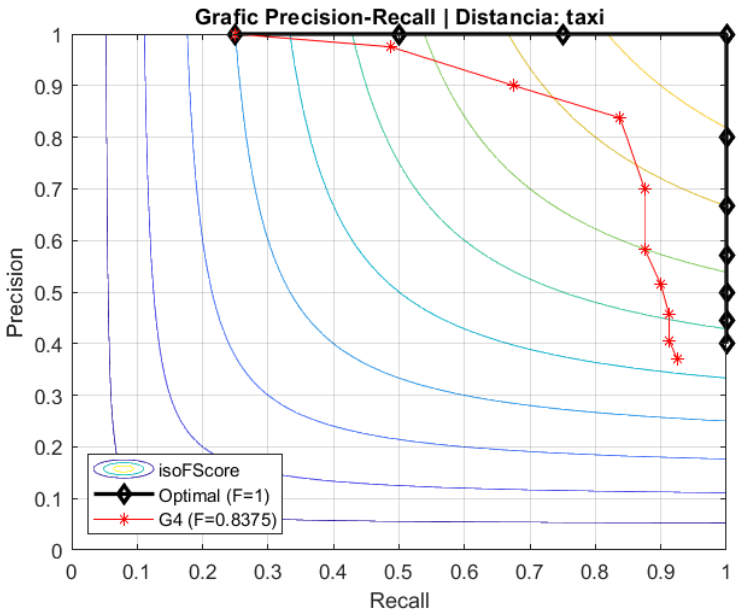
- "L1" (també coneguda com "Taxi" o "Manhattan")
- Hellinger
- MAE
- Chi Quadrat

S'ha avaluat la seva utilitat segons l'F-score obtingut per cada distància:




Com es pot apreciar als gràfics, les distàncies L1, MAE i Chi Quadrat donen una F-score de 0,8375 mentre que la Hellinger proporciona una petita millora, arribant a 0,8625.

Tot i que les tres primeres tenen la mateixa F-score, amb la Chi Quadrat el sistema sempre és capaç de trobar correctament la segona imatge mentres que la L1 i la MAE sembla que "casi" sempre ho aconsegueixen.

Finalment, amb la distància Hellinger sempre es troba correctament la segona imatge i casi que també la tercera.

Es per això i per l'F-score que proporciona la distància Hellinger que es va decidir com a distància base per la primera versió del sistema. També ha sigut la distància que s'ha utilitzat per a totes les posteriors configuracions.

## Configuració "256_base" vs altres configuracions

A partir de la configuració "256_base" s'han desenvolupat les següents configuracions. Totes pengen d'aquesta configuració inicial.

Es procedeix a explicar breument cada configuració. Al final d'aquest apartat es comparen amb una taula resum comparant resultats obtinguts.

## Configuració "256_opt"

Aquesta configuració sorgeix al veure l'elevat temps d'execució de la primera configuració. L'objectiu era reduir temps d'execució però mantenint la mateixa F-score.

Per intentar reduir el temps de processat de les 2000 imatges es van desenvolupar les funcions `prog2_rgb2hmmd_vect.m` i `prog2_hmmd_quantize_opt.m`, que intenten ser versions optimitzades de les corresponents funcions originals de la configuració base.

Les optimitzacions son utilitzar operacions vectorials en comptes de pixel a pixel, i funcions més comprimides, amb menys instruccions.

Seguidament es van substituir a les corresponents seccions del bucle principal en `Prog2_CSD_Feature_Extraction_opt.m`

Es va aconseguir reduir el temps d'execució de 2 hores i 50 minuts a aproximadament 2 hores:

```
>> Prog2_CSD_Feature_Extraction_opt
Hora inicio procesado optimizada 2000 imagenes con 256 bins: 07/05/25-18:48
Elapsed time is 7387.701351 seconds.
Hora finalizacion procesado opt 2000 imagenes con 256 bins: 07/05/25-20:52
>>
```

7387 / 60 = 123,11 minuts => 2 horas y 3 minutos

Es va aconseguir mantenir la mateixa F-score de 0,8625

Per veure el codi complert revisar l'annex A1.3.

## Configuració “256_delmat”

L’estàndard defineix que “*les imatges que no son d’un tamany nominal han de ser sub-mostrejades (delmades), tant verticalment com horitzontalment, per reduir el cost computacional*”.

El factor de delmat es donat per $K = 2^p$ on

$$p = max\{0, floor(log_2(\sqrt{W \cdot H}) - 7,5)\}$$

En el nostre cas, on totes les imatges son de tamany 640x480, p=1 i per tant el factor de delmat K = 2.

En aquesta configuració s’ha implementat el delmat a l’inici del bucle principal `Prog2_CSD_Feature_Extraction_delmat.m`. Per a fer-ho, abans s’han filtrat els canals RGB de la imatge original per evitar problemes de aliasing.

Al executar aquesta configuració, el temps d’execució va baixar dràsticament:

```
>> Prog2_CSD_Feature_Extraction_delmat
Hora inicio procesado pixel a pixel con delmado 2000 imagenes con 256 bins: 09/05/25-10:19
Elapsed time is 2671.320507 seconds.
Hora finalizacion procesado delmat 2000 imagenes con 256 bins: 09/05/25-11:04
>>
```

2671 / 60 = 44,5 minuts

I per la nostra sorpresa, la F-score va pujar fins a 0,9.
S’adjunta a l’Annex 3 el fitxer `output4_256_delmat.txt` que genera aquesta F-score.

El codi complet del bucle principal modificat es pot consultar a l’annex A1.4.

## Confifuració “256_4x4”

L’estàndard indica que el tamany òptim, el que proporciona millors resultats, de l’element estructurant es de 8x8.

Aquesta configuració sorgeix de voler comprovar aquesta afirmació. Per això, és modifica el bucle principal i es genera el fitxer `Prog2_CSD_Feature_Extractio_bola4x4n.m` en el que el tamany de l’element estructurant passa de 8x8 a 4x4.

Esperavem que el temps d’execució incrementés bastant respecte la configuració “256_base”. Per a la nostra sorpresa va disminuir uns 15 minuts:

```
>> Prog2_CSD_Feature_Extractio_bola4x4n
Hora inicio procesado pixel a pixel bola 4x4 2000 imagenes con 256 bins: 10/05/25-10:28
Elapsed time is 9411.199550 seconds.
Hora finalizacion procesado pixel a pixel bola 4x4 2000 imagenes con 256 bins: 10/05/25-13:05
>>
```

9411,2 / 60 = 156 minuts = 2 hores 36 minuts

I, tal i com esperàvem, la F-score va baixar a 0,85
El codi complet del bucle principal modificat es pot consultar a l’annex A1.5.

## Configuració “128”

Finalment, la última configuració tracta de treballar amb la meitat de bins per veure si es poden obtenir millors resultats.

Es podria haver agafat la base de dades original i haver fet “bin unification” tal i com indica l’estàndard, però vàrem decidir modificar la configuració inicial de tal forma que sigui com si haguessim optat per aquest numero de bins des d’un inici.

Per això, es modifica i genera la funció `prog2_hmmd_quantize_128.m` i s’incorpora al nou bucle principal `Prog2_CSD_Feature_Extraction_128.m`

Aquesta configuració triga un temps d’execució similar a la configuració base:

```
>> Prog2_CSD_Feature_Extraction_128
Hora inicio procesado pixel a pixel 2000 imagenes con 128 bins: 10/05/25-14:04
Elapsed time is 9805.335634 seconds.
Hora finalizacion procesado pixel a pixel 2000 imagenes con 128 bins: 10/05/25-16:48
>>
```

9805/60 = 163 minuts => 2 hores i 43 minuts

I vem aconseguir una F-score lleugerament inferior a la configuració base, de 0,85.

El codi complet del bucle principal modificat i la nova funció es pot consultar a l’annex A1.6.

## Taula resum i comparativa de resultats

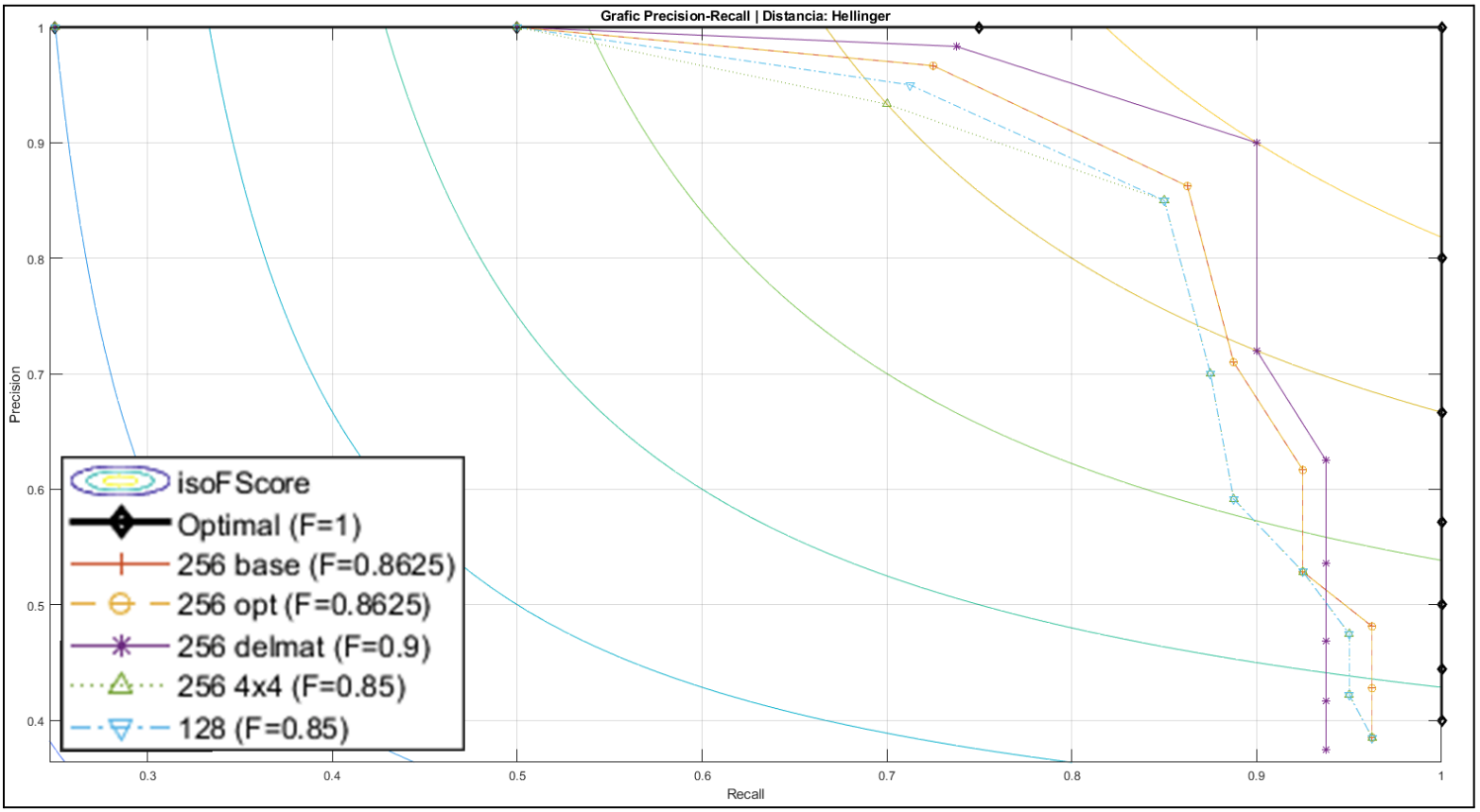
A continuació es comparen les diferents configuracions i els resultats obtinguts:

| Configuració | Nombre bins | Funcions optimitzades? | Tamany element estructurant | Aplica delmat? | Temps d’execució | F-score obtinguda |
|---|---|---|---|---|---|---|
| 256_base | 256 | No | 8x8 | No | 2h 50m | 0,8625 |
| 256_opt | 256 | Si | 8x8 | No | 2h 3m | 0,8625 |
| 256_delmat | 256 | No | 8x8 | Si | 44m | 0,9 |
| 256_4x4 | 256 | No | 4x4 | No | 2h 36m | 0,85 |
| 128 | 128 | No | 8x8 | No | 2h 43 | 0,85 |

Totes les configuracions han sigut executades amb el portàtil de l’Alejandro, un Lenovo Thinkpad E495. Es a dir, tots els temps d’execució esmentats son d’aquest portàtil, que té la següent funció “benchmark” de Matlab.

| | | | | | | |
|---|---|---|---|---|---|---|
| This machine | 2.0888 | 0.8079 | 0.2732 | 0.9549 | 0.7233 | 0.5486 |

Gràfic comparatiu:




Totes les configuracions es comporten de la mateixa forma fins que arriben a la tercera imatge.

La configuració “256_4x4” es la que presenta pitjors resultats a la 3a imatgei, però a la 4a imatge segueix el mateix comportament que la configuració “128”, ambdues amb una F-score final de 0,85.

La configuració “256_base” i “256_opt” tenen la mateixa corba precision-recall (estan solapades), amb una F-score de 0,8625. Es un resultat esperable donada que la configuració “opt” només es centrava en millorar temps d’execució.

Finalment, la corba amb millors resultats es la de la configuració “256_delmat”, arribant a proporcionar una f-score de 0,9.

Aquesta f-score de 0,9 al principi ens va sorprendre inicialment. Però si l’estàndard defineix el CSD amb un tamany d’imatge nominal i ja indica que, en cas de processar imatges de diferent tamany cal fer un delmat amb factor K, tindria sentit que la configuració “256_base” doni una F-score inferior a la “256_delmat”.
És a dir, la configuració “256_delmat” es la que més s’apropa a l’estàndard i per això dona els millors resultats, tant en temps d’execució com de F-score.

L’script `Prog2_Comparativa.m` que genera aquest gràfic es pot consultar a l’annex A1.7.

S'adjunta el següent enllaç de Drive amb tots els fitxers generats i relacionats amb el aquest projecte:
https://drive.google.com/drive/folders/1Jok--FkVXsJAbKme6ZonEZ5wwdyoqKDh?usp=sharing

## Millores per versions futures

A nivell millorar resultats, la F-score, dubtem que es puguin aplicar masses millores més. Hem arribat a la conclusió de que la configuració "256_delmat" es la que més s'apropa a l'estàndard i és per això que proporciona millors resultats.

L'únic que se'ns ocorre per intentar millorar resultats és explorar la via de la quantització no lineal que es fa just al final del procés. Per exemple, es podria provar amb diferents valors del paràmetre "a".

A nivell de temps d'execució, abans de desenvolupar la configuració "256_delmat" ens haviem centrat en optimitzar codi. Per aquesta via hem aconseguit "retallar" ben bé 50 minuts respecte la configuració base, una millora significativa.

A continuació vàrem implementar la configuració que aplica delmat i per la nostra sorpresa la millora de temps d'execució va ser dràstica, passant de 2 hores 50 minuts a 44 minuts. Ja ens sembla un temps d'execució prou baix, però podriem intentar implementar les funcions optimitzades en aquesta configuració i veure quan triga.

Finalment, a nivell d'estructuració del projecte es podrien millorar moltes coses.
De la forma en que s'ha treballat s'han acabat generant molts scripts molt similars entre ells, quan en realitat es podrien unificar en un sol i gestionar-los amb paràmetres. En cas de seguir treballant amb aquest projecte una millora de "quality of life" seria el unificar configuracions en un mateix script i gestionar-les amb variables "flag".

## Dificultats

Una de les majors dificultats ha sigut el saber si estavem desenvolupant correctament el descriptor. És a dir, el tenir la certesa de desenvolupar de forma correcta un algorisme que ni hem pensat nosaltres i que tampoc acabem d'entendre al 100%, ni perquè fa el que fa.

El CSD requereix que les imatges estiguin en l'espai de color HMMD, però Matlab no proporciona cap funció del tipus "rgb2hmmd()". Per a tenir la certesa de que estàvem començant a programar correctament vam desenvolupar la nostra funció `prog2_rgb2hsv` i la vem comparar amb la que proporciona Matlab.

Al saber que havíem desenvolupat correctament la funció `rgb2hsv` vem implementar la funció `prog2_rgb2hmmd`. Després, per saber si l'histograma CSD inicial estava ben fet, vem preparar algunes imatges molt senzilles per fer-ho nosaltres a mà i comparar resultats amb les versions inicials de la configuració "256_base".

La funció esmentada i aquestes comprovacions inicials es poden trobar a l'Annex 4.

Finalment, va arribar un punt on no podíem comprovar manualment les fotos més complexes, ni les operacions més complicades, i “vem tirar endavant” fins poder observa resultats de f-score amb la primera configuració. L’haver obtingut primerament una f-score de 0,83 (amb configuració base i distància L1) ja ens va ser indicador de que anàvem per bon camí.

Si haguessim obtingut una f-score de 0,5 +- 0,1 (propera a la de la primera pràctica, amb nivells de gris) si que ens hauria fet plantejar que una part crítica de lo que s’havia desenvolupat estava fundamentalment malament.

# Annex 1: Codi complert

## A1.1: Bucle principal i funcions de la configuració “256_base”

### Bucle principal `Prog2_CSD_Feature_Extraction.m`

```
clear all
% Define Image Database directory and images filename prefix
ImDB_path = [pwd,'\UKentuckyDatabase\'];
ImTest_path = [pwd,'\img\'];
ImDB_name_prefix = 'ukbench';
% Parameters for feature extraction
hist_bins = 256;
num_images = 2000;
H_csd = zeros(num_images,hist_bins);
disp(['Hora inicio procesado pixel a pixel ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
tic

%% Inici bucle principal
for i = 0:(num_images-1)
   filename = [ImDB_path, ImDB_name_prefix, sprintf('%05d.jpg',i)];
   %% 1.1 Conversió a l'espai de color HMMD
   mode = 'HDS'; %mode pot ser HMM o HDS

   % conversio pixel a pixel (amb "prog2_rgb2hmmd.m")
   I_hmmd = prog2_rgb2hmmd(imread(filename),mode);

   %% 1.2 Quantitzacio no uniforme de la imatge hmmd

   % Cuantizacion por pixel (usando "prog2_hmmd_quantize.m")
   I_files = size(I_hmmd,1);
   I_columnes = size(I_hmmd,2);
   I_hmmd_quantized = zeros(I_files,I_columnes);
   for ii = 1:(I_files)
       for jj = 1:I_columnes

I_hmmd_quantized(ii,jj)=prog2_hmmd_quantize(I_hmmd(ii,jj,1),I_hmmd(ii,jj,2),I_hmmd(ii,jj,3));
       end
   end

   %% 2 Escombrat de la imatge amb la bola 8x8

   %Cada posicio "k" del csd_hist representa el numero de boles 8x8 en las que
   %apareix al menys una vegada el bin "k"
   csd_hist = zeros(1, 256);  % Inicializa histograma

   tamany_bola = 8;

   I_q = I_hmmd_quantized;
   I_files = size(I_q,1);
   I_columnes = size(I_q,2);

   for kk = 1:(I_files - tamany_bola + 1)
       for ll = 1:(I_columnes - tamany_bola + 1)

           % Extraer bloque 8x8 con solapamiento
```

```
            block = I_q(kk:kk+tamany_bola-1, ll:ll+tamany_bola-1);

            % Encuentra los bins únicos presentes en el bloque
            bins_present = unique(block);

            % Incrementa el histograma una vez por bin presente en el bloque
            csd_hist(bins_present + 1) = csd_hist(bins_present + 1) + 1;
        end
    end


    %% 3 Normalitzacio i quantitzacio no lineal

    % normalitzacio
    csd_hist = csd_hist / sum(csd_hist);

    % quantitzacio no lineal
    a = 0.4;
    csd_quantized = round( (log(1 + a * csd_hist) / log(1 + a)) * 255 );
    H_csd(i+1,:) = csd_quantized;
end
toc
disp(['Hora finalizacion procesado pixel a pixel ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
save('H_csd_2000.mat',"H_csd")
```

## Funció `prog2_rgb2hmmd.m`

```
function [hmmd_image_out] = prog2_rgb2hmmd(rgb_image_in,mode)
% mode pot ser 'HMM' per a (Hue,Max,Min) o 'HDS' per a (Hue,Diff,Sum).
% Per defecte es HDS.
% Implementacio de la funcio rgb2hmmd
% La entrada es una imatge de l'espai de color RGB
% La sortida es la imatge d'entrada convertida a l'espai de color HMMD
if nargin < 2
    mode = 'HDS';
else
    % Si el mode no es HMM o el mode no es HDS, el mode sera HMM
    if ~strcmp(mode,'HMM') && ~strcmp(mode,'HDS')
        mode = 'HDS';
    end
end
hmmd_image_out = zeros(size(rgb_image_in,1),size(rgb_image_in,2),3);
% Es normalitza la imatge d'entrada en cas de que no ho estigui ja
if isa(rgb_image_in, 'uint8')
    rgb_image_in = double(rgb_image_in) / 255;
end
for i = 1:size(rgb_image_in,1)
    for j = 1:size(rgb_image_in,2)
        % Calcul de Max i Min
        maxx=max(rgb_image_in(i,j,:));
        minn=min(rgb_image_in(i,j,:));

        % Calcul del Hue
        r = rgb_image_in(i,j,1);
        g = rgb_image_in(i,j,2);
        b = rgb_image_in(i,j,3);
        if maxx == minn
```

```
            hue = 0;
        elseif (maxx == r) && (g >= b)
            hue = 60*((g-b)/(maxx-minn));
        elseif (maxx == r) && (g < b)
            hue = 360 + 60*((g-b)/(maxx-minn));
        elseif g == maxx
            hue = 60*(2.0+ ((b-r)/(maxx-minn)));
        else
            hue = 60*(4.0+ ((r-g)/(maxx-minn)));
        end
        hue = hue / 360; % Es normalitza el valor de Hue
        %Calcul del Difference
        difference = maxx - minn;
        %calcul de sum
        summ = (maxx + minn)/2;
        % Assignacio final segons el mode
        if strcmp(mode,'HMM')
            hmmd_image_out(i,j,1)=hue;
            hmmd_image_out(i,j,2)=maxx;
            hmmd_image_out(i,j,3)=minn;
        else
            hmmd_image_out(i,j,1)=hue;
            hmmd_image_out(i,j,2)=difference;
            hmmd_image_out(i,j,3)=summ;
        end
    end
end
end
```

## Funció `prog2_hmmd_quantize.m`

```
function bin = prog2_hmmd_quantize(H, D, S)
% Donat un pixel HMMD (H,D,S) (normalitzat, de 0 a 1) aquesta funció
% retorna un numero entre 0 i 255, indicant el bin del histograma al qual
% pertany.
diff_thresholds = [6, 20, 60, 110] / 255;
%         Hue  Sum
levels = [ 1, 32;  % Subespai 0 - 1*32=32 + 0 = 32 | 1r offset
           4,  8;  % Subespai 1 - 4*8=32 + 32 = 64 | 2n offset
          16,  4;  % Subespai 2 - 16*4=64 + 64 = 128 | 3r offset
          16,  4;  % Subespai 3 - 16*4=64 + 128 = 192 | 4t offset
          16,  4]; % Subespai 4 - 16*4=64 + 192 = 256
% Offset per a cada zona de l'histograma
zone_offsets = [0, 32, 64, 128, 192];  % acumulacio de bins per subespai
% Deteccio de subespai
zone = find(D < diff_thresholds, 1);
if isempty(zone)
    zone = 5;
end
hue_levels = levels(zone,1);
sum_levels = levels(zone,2);
% Evitar desbordaments per valors = 1.0
epsilon = 1e-10;
H = min(H, 1 - epsilon);
S = min(S, 1 - epsilon);
% Quantització
h_q = floor(H * hue_levels);
s_q = floor(S * sum_levels);
```

```
% Índex global del bin
bin = zone_offsets(zone) + s_q * hue_levels + h_q;
end
```

## A1.2: Bucle principal i funcions del sistema d'avaluació de resultats

### Bucle principal Prog2.m

```
clear all;
tic
%% Bases dedades:
% 256_base: La primera que es va fer i amb la que es va generar "output4.txt"
% 256_opt: La que es va generar després d'optimitzar els metodes de conversió i
quantificació
% 256_delmat: La que es va generar després d'aplicar delmat de factor K
% 256_4x4 :La que es va generar al aplicar una bola 4x4 en comptes de 128
% 128: La que es va generar quantitzant a 128 bins en comptes de 256
BBDD = "256_base";
if strcmp(BBDD,"256_base")
   load("H_csd_2000.mat","H_csd") % Carreguem la BDD d'histogrames

elseif strcmp(BBDD,"256_opt")
   load("H_CSD_2000imgs_256bins_opt.mat","H_csd_opt_2000")
   H_csd = H_csd_opt_2000;
elseif strcmp(BBDD,"256_delmat")
   load("H_CSD_2000imgs_256bins_delmat.mat","H_CSD_delmat")
   H_csd = H_CSD_delmat;
elseif strcmp(BBDD,"256_4x4")
   load("H_CSD_bola4x4_2000imgs_256bins_pixelApixel.mat","H_csd_bola4x4")
   H_csd = H_csd_bola4x4;
else
   load("H_CSD_2000imgs_128bins_pixelApixel.mat","H_csd_128")
   H_csd = H_csd_128;
end
% Data variables LAB
% User_root      = 'C:\Users\biel.sala';
% Data_root      = '\Downloads\Prog1\';
% Input_filename = 'input.txt';
% Output_filename= 'output4.txt';
% Candidates     = 10;  % Number of candidates to retrieve
% Data variables laptop Alejandro
User_root      = 'G:\Mi unidad\UPC';
Data_root      = '\Curso\10 - 24-25 P\PIV\PIV Lab\Prog2\';
Input_filename = 'input.txt';
Output_filename= 'output.txt';
Candidates     = 10;  % Number of candidates to retrieve
tipus_distancia = 'hellinger'; % 'taxi', 'hellinger', 'mae' o 'chi2'
num_bdd_images=size(H_csd,1);
% Define Image Database directory and images filename prefix
picture_path = 'G:\Mi unidad\UPC\Curso\10 - 24-25 P\PIV\PIV
Lab\Prog2\UKentuckyDatabase\';
fid = fopen(Input_filename, 'r'); % Abrir el archivo en modo lectura
Input = textread([User_root, Data_root, Input_filename],'%s');
% Get the Number of images to be Queried
num_input_images = length(Input);
```

```
%
noms = readlines([User_root, Data_root, Input_filename]);
% Open output file for writing the results
a=fopen([User_root, Data_root, Output_filename],'w');
perfect_precision = [1,1,1,1,4/5,4/6,4/7,4/8,4/9,4/10];
perfect_recall = [0.25,0.5,0.75,1,1,1,1,1,1,1];
precision = zeros(num_input_images,Candidates);
recall = zeros(num_input_images,Candidates);
for i=1:num_input_images
   % Per cada imatge a input.txt busquem el seu histograma
   picture_name = noms(i);
   %Extraccio del numero de la imatge que estem fent query
   auxStr=extractBefore(picture_name,".jpg");
   auxStr2 =extractAfter(auxStr,'ukbench');
   picture_name_number=str2num(auxStr2);
   %"agafem" l'histograma de la imatge que estem query de la BBDD.
   % +1 perquè les imatges comencem per la 00000 però l'index Matlab
   % comença pel numero 1
   picture_hist = H_csd(picture_name_number+1,:);

   %Comparem l'histograma calculat amb tots els histogramas de H
   for j=1:num_bdd_images
       d(j)=distancia4_variable(picture_hist,H_csd(j,:),tipus_distancia);
   end

   fprintf(a,'Retrieved list for query image %s \n',char(Input(i)));

   % Here, is suposed that the system gets a index vector to the most similar
   % images in the vector Similar_images;
   Similar_images = buscador_10index_minims(d);
   % Writing the results at the Output file
   for j=1:Candidates
       fprintf(a,'%s\n',sprintf('ukbench%05d.jpg',Similar_images(j)));
   end
   fprintf(a,'\n');
   %Comptar cuants encert hi ha hagut per cada imatge per fer el
   aciertos = 0;
   for k=1:length(Similar_images)
       %disp("Comparem " + picture_name_number + " amb " + Similar_images(k));
       if (cuentaAciertos(picture_name_number,Similar_images(k)))
           aciertos = aciertos +1;
       end
       precision(i,k) = aciertos/k;
       recall(i,k) = aciertos/4;
   end
end
%% Calcul f_score
%promitjar tots els precision i recall i calcular l'fscore
precision_avg = sum(precision)/num_input_images;
recall_avg = sum(recall)/num_input_images;
f_score = max((2.*precision_avg.*recall_avg)./(precision_avg+recall_avg));
%% gráfico del precision recall
% contour per les isolines
x = linspace(0,1);
y = linspace(0,1);
```

```
[X,Y] = meshgrid(x,y); % contour nomes accepta matrius
Z = (2.*X.*Y)./(X+Y); % formula del f-score per a que calculi les isolinies
contour(X,Y,Z)
hold on
plot(perfect_recall,perfect_precision,'k-diamond','LineWidth',2)
grid on
hold on
title(['Grafic Precision-Recall | Distancia: ', tipus_distancia])
plot(recall_avg,precision_avg,'r-*')
xlabel('Recall');
ylabel('Precision');
legend('isoFScore','Optimal (F=1)',strcat('G4
(F=',num2str(f_score),')'),'Location','southwest')
toc
save("recall_" + BBDD,"recall_avg");
save("precision_" + BBDD,"precision_avg");
```

### Funció `distancia4_variable.m`

```
function [distancia] = distancia4_variable(h1,h2,tipus)
% Es una metrica de similitud. Quant menor sigui el valor 'distancia',
més s'asemblen
% els histogrames 'h1' i 'h2'.
%es normalitzen els histogrames:
h1 = h1 / sum(h1);
h2 = h2 / sum(h2);
if strcmp(tipus,'taxi') % Distancia L1, Taxicab o Manhattan:
   distancia = sum(abs(h1-h2));
elseif strcmp(tipus,'hellinger')
   distancia = sqrt(0.5) * sqrt(sum((sqrt(h1) - sqrt(h2)).^2));
elseif strcmp(tipus,'mae')
   distancia = mean(abs(h1 - h2));
elseif strcmp(tipus,'chi2')
   distancia = sum(((h1 - h2).^2) ./ (h1 + h2 + eps));
else
   distancia = sum(abs(h1-h2));
end

end
```

### Funció `buscador_10index_minims.m`

```
function [outputArg1] = buscador_10index_minims(input)
% Ordena les distancies calculades i retorna els indexs de les 10
imatges mes similars
   input_sorted = sort(input);

   for i=1:10
```

```
        indexes(i) = find(input == input_sorted(i),1,'last');
    end
    outputArg1 = indexes -ones(1,10);
end
```

## Funció `cuentaAciertos.m`

```
function [output] = cuentaAciertos(numImgRef, numImgTest)
%CUENTAACIERTOS Summary of this function goes here
%  Se li pasen dos dumeros de imatge
%  Ha de mirar si la imatge input2 es un acierto de la imatge inpu1
%Mitjançant aritmetica modular, es calcula quin es el numero inicial del
llistat de la
%imatge de Referencia per a despres fer un vector dels numeros que es
%consideren aciertos
numPrimeraImatge = numImgRef - mod(numImgRef,4);
imatgesAcierto = [numPrimeraImatge,numPrimeraImatge+1,numPrimeraImatge+2,numPrimeraImatge+3 ];
% es comprova si el numero de la imatge a comparar (ImgTest) forma part del
%vector
output = ismember(numImgTest,imatgesAcierto);
end
```

## A1.3 Bucle principal i funcions modificades de la configuració “256_opt”

### Bucle principal `Prog2_CSD_Feature_Extraction_opt.m`

```
clear all
% Feature extraction de 256 bins pero amb una conversió RGV a HMMD  vectorial i una funcio de
cuantificació optimitzada
% Define Image Database directory and images filename prefix
ImDB_path = [pwd,'\UKentuckyDatabase\'];
ImTest_path = [pwd,'\img\'];
ImDB_name_prefix = 'ukbench';
% Parameters for feature extraction
hist_bins = 256;
num_images = 2000;
H_csd_opt_2000 = zeros(num_images,hist_bins);
disp(['Hora inicio procesado optimizada ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
tic
%% Inici bucle principal
for i = 0:(num_images-1)
   filename = [ImDB_path, ImDB_name_prefix, sprintf('%05d.jpg',i)];
   %% 1.1 Conversió a l'espai de color HMMD
   mode = 'HDS'; %mode pot ser HMM o HDS
   % conversio matricial (amb "prog2_rgb2hmmd_vect.m")
   I_rgb = imread(filename);
   I_hmmd = prog2_rgb2hmmd_vect(I_rgb, mode);

   %% 1.2 Quantitzacio no uniforme de la imatge hmmd
   % cuantizacion por matriz (usando "prog2_hmmd_quantize_vect.m")
   I_files = size(I_hmmd,1);
   I_columnes = size(I_hmmd,2);
   I_hmmd_quantized = zeros(I_files,I_columnes);
   for ii = 1:(I_files)
       for jj = 1:I_columnes

I_hmmd_quantized(ii,jj)=prog2_hmmd_quantize_opt(I_hmmd(ii,jj,1),I_hmmd(ii,jj,2),I_hmmd(ii,jj,3));
       end
   end


   %% 2 Escombrat de la imatge amb la bola 8x8

   %Cada posicio "k" del csd_hist representa el numero de boles 8x8 en las que
   %apareix al menys una vegada el bin "k"
   csd_hist = zeros(1, 256);  % Inicializa histograma

   tamany_bola = 8;

   I_q = I_hmmd_quantized;
   I_files = size(I_q,1);
   I_columnes = size(I_q,2);

   for kk = 1:(I_files - tamany_bola + 1)
       for ll = 1:(I_columnes - tamany_bola + 1)

           % Extraer bloque 8x8 con solapamiento
```

```
            block = I_q(kk:kk+tamany_bola-1, ll:ll+tamany_bola-1);

            % Encuentra los bins únicos presentes en el bloque
            bins_present = unique(block);

            % Incrementa el histograma una vez por bin presente en el bloque
            csd_hist(bins_present + 1) = csd_hist(bins_present + 1) + 1;
        end
    end


    %% 3 Normalitzacio i quantitzacio no lineal

    % normalitzacio
    csd_hist = csd_hist / sum(csd_hist);

    % quantitzacio no lineal
    a = 0.4;
    csd_quantized = round( (log(1 + a * csd_hist) / log(1 + a)) * 255 );
    H_csd_opt_2000(i+1,:) = csd_quantized;
end
toc
disp(['Hora finalizacion procesado opt ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
save(['H_CSD_',num2str(num_images),'imgs_',num2str(hist_bins),'bins_opt'],"H_csd_opt_2000")
```

## Funció `prog2_rgb2hmmd_vect.m`

```
function hmmd_image_out = prog2_rgb2hmmd_vect(rgb_image_in, mode)
% Versió vectoritzada de RGB a HMMD (HMM o HDS)
% Retorna una imatge de la mateixa mida amb 3 canals: (H, Max+Min o Diff, Min o Sum)
if nargin < 2
    mode = 'HDS';
end
% Si RGB està en uint8, normalitza a [0,1]
if isa(rgb_image_in, 'uint8')
    rgb_image_in = double(rgb_image_in) / 255;
end
R = rgb_image_in(:,:,1);
G = rgb_image_in(:,:,2);
B = rgb_image_in(:,:,3);
MaxVal = max(cat(3,R,G,B), [], 3);
MinVal = min(cat(3,R,G,B), [], 3);
Diff = MaxVal - MinVal;
Sum  = (MaxVal + MinVal) / 2;
Hue = zeros(size(R));
mask = (Diff ~= 0);  % on hi ha diferència entre canals
% Càlcul de Hue segons HSV
R_m = R(mask); G_m = G(mask); B_m = B(mask); D_m = Diff(mask); Mx = MaxVal(mask);
hue = zeros(size(R_m));
% Condicions vectoritzades
condR_Gb = (Mx == R_m) & (G_m >= B_m);
condR_Gl = (Mx == R_m) & (G_m < B_m);
condG = (Mx == G_m);
condB = (Mx == B_m);
hue(condR_Gb) = 60 * (G_m(condR_Gb) - B_m(condR_Gb)) ./ D_m(condR_Gb);
```

```
hue(condR_Gl) = 360 + 60 * (G_m(condR_Gl) - B_m(condR_Gl)) ./ D_m(condR_Gl);
hue(condG)    = 60 * (2 + (B_m(condG) - R_m(condG)) ./ D_m(condG));
hue(condB)    = 60 * (4 + (R_m(condB) - G_m(condB)) ./ D_m(condB));
Hue(mask) = hue / 360;  % normalitzar a [0,1]
% Assignació final segons mode
hmmd_image_out = zeros(size(rgb_image_in));
hmmd_image_out(:,:,1) = Hue;
if strcmp(mode, 'HMM')
   hmmd_image_out(:,:,2) = MaxVal;
   hmmd_image_out(:,:,3) = MinVal;
else  % HDS
   hmmd_image_out(:,:,2) = Diff;
   hmmd_image_out(:,:,3) = Sum;
end
end
```

## Funció `prog2_hmmd_quantize_opt.m`

```
function bin = prog2_hmmd_quantize_opt(H, D, S)
% Versió optimitzada de la quantificació HMMD
% Entrada: H, D, S ∈ [0, 1]
% Sortida: bin ∈ [0, 255]
% Constants precalculades
diff_thresholds = [6, 20, 60, 110] / 255;
zone_offsets = [0, 32, 64, 128, 192];
levels = [ 1, 32; 4, 8; 16, 4; 16, 4; 16, 4 ];
% Determinar zona sense usar 'find'
zone = sum(D >= diff_thresholds) + 1;
% Obtenir els nivells corresponents
hue_levels = levels(zone, 1);
sum_levels = levels(zone, 2);
% Truncament ràpid sense funció 'min'
H = H - (H == 1) * eps;
S = S - (S == 1) * eps;
% Quantificació
h_q = floor(H * hue_levels);
s_q = floor(S * sum_levels);
% Càlcul del bin final
bin = zone_offsets(zone) + s_q * hue_levels + h_q;
end
```

## A1.4 Bucle principal modificat de la configuració “256_delmat”

### Bucle principal `Prog2_CSD_Feature_Extraction_delmat.m`

```
clear all
% Define Image Database directory and images filename prefix
ImDB_path = [pwd,'\UKentuckyDatabase\'];
ImTest_path = [pwd,'\img\'];
ImDB_name_prefix = 'ukbench';
% Parameters for feature extraction
hist_bins = 256;
num_images = 2000;
H_CSD_delmat = zeros(num_images,hist_bins);
disp(['Hora inicio procesado pixel a pixel con delmado ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
tic
% El estandar MPEG-7 defineix el factor de delmat o "subsampling" K = 2^p
% on p = max{0, floor(log2(sqrt(W*H))-7.5)}
% Totes les imatges son 640x480
W=640;
H=480;
p = floor(log2(sqrt(W*H))-7.5);
K = 2^p;
%% Inici bucle principal
for i = 0:(num_images-1)
   filename = [ImDB_path, ImDB_name_prefix, sprintf('%05d.jpg',i)];
   %% 1.1 Conversió a l'espai de color HMMD
   mode = 'HDS'; %mode pot ser HMM o HDS
   % Leer imagen
   I_rgb = imread(filename);

   % Convertir a double para precisión en el filtrado
   I_rgb = im2double(I_rgb);

   % Definir un filtro pasabajo
   h = fspecial('gaussian', [5 5], 1);  % sigma = 1

   % Aplicar el filtro a cada canal RGB
   I_filt(:,:,1) = imfilter(I_rgb(:,:,1), h, 'replicate');
   I_filt(:,:,2) = imfilter(I_rgb(:,:,2), h, 'replicate');
   I_filt(:,:,3) = imfilter(I_rgb(:,:,3), h, 'replicate');
   % delmat con paso K
   I_subsampled = I_filt(1:K:end, 1:K:end, :); % Selecciona una de cada K filas y columnas

   % Conversión RGB a HMMD
   I_hmmd = prog2_rgb2hmmd(I_subsampled, mode); % Usa aquí la imagen subsampleada
   %% 1.2 Quantitzacio no uniforme de la imatge hmmd

   % Cuantizacion por pixel (usando "prog2_hmmd_quantize.m")
   I_files = size(I_hmmd,1);
   I_columnes = size(I_hmmd,2);
   I_hmmd_quantized = zeros(I_files,I_columnes);
   for ii = 1:(I_files)
       for jj = 1:I_columnes

I_hmmd_quantized(ii,jj)=prog2_hmmd_quantize(I_hmmd(ii,jj,1),I_hmmd(ii,jj,2),I_hmmd(ii,jj,3));
       end
```

```
    end

    %% 2 Escombrat de la imatge amb la bola 8x8

    %Cada posicio "k" del csd_hist representa el numero de boles 8x8 en las que
    %apareix al menys una vegada el bin "k"
    csd_hist = zeros(1, 256);  % Inicializa histograma

    tamany_bola = 8;

    I_q = I_hmmd_quantized;
    I_files = size(I_q,1);
    I_columnes = size(I_q,2);

    for kk = 1:(I_files - tamany_bola + 1)
        for ll = 1:(I_columnes - tamany_bola + 1)

            % Extraer bloque 8x8 con solapamiento
            block = I_q(kk:kk+tamany_bola-1, ll:ll+tamany_bola-1);

            % Encuentra los bins únicos presentes en el bloque
            bins_present = unique(block);

            % Incrementa el histograma una vez por bin presente en el bloque
            csd_hist(bins_present + 1) = csd_hist(bins_present + 1) + 1;
        end
    end


    %% 3 Normalitzacio i quantitzacio no lineal

    % normalitzacio
    csd_hist = csd_hist / sum(csd_hist);

    % quantitzacio no lineal
    a = 0.4;
    csd_quantized = round( (log(1 + a * csd_hist) / log(1 + a)) * 255 );
    H_CSD_delmat(i+1,:) = csd_quantized;
end
toc
disp(['Hora finalizacion procesado delmat ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
save(['H_CSD_',num2str(num_images),'imgs_',num2str(hist_bins),'bins_delmat'],"H_CSD_delmat")
```

## A1.5 Bucle principal modificat de la configuració “256_4x4”

### Bucle principal `Prog2_CSD_Feature_Extractio_bola4x4n.m`

```
clear all
% Define Image Database directory and images filename prefix
ImDB_path = [pwd,'\UKentuckyDatabase\'];
ImTest_path = [pwd,'\img\'];
ImDB_name_prefix = 'ukbench';
% Parameters for feature extraction
hist_bins = 256;
num_images = 2000;
H_csd_bola4x4 = zeros(num_images,hist_bins);
disp(['Hora inicio procesado pixel a pixel bola 4x4 ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
tic
%% Inici bucle principal
for i = 0:(num_images-1)
   filename = [ImDB_path, ImDB_name_prefix, sprintf('%05d.jpg',i)];
   %% 1.1 Conversió a l'espai de color HMMD
   mode = 'HDS'; %mode pot ser HMM o HDS

   % conversio pixel a pixel (amb "prog2_rgb2hmmd.m")
   I_hmmd = prog2_rgb2hmmd(imread(filename),mode);

   %% 1.2 Quantitzacio no uniforme de la imatge hmmd

   % Cuantizacion por pixel (usando "prog2_hmmd_quantize.m")
   I_files = size(I_hmmd,1);
   I_columnes = size(I_hmmd,2);
   I_hmmd_quantized = zeros(I_files,I_columnes);
   for ii = 1:(I_files)
       for jj = 1:I_columnes

I_hmmd_quantized(ii,jj)=prog2_hmmd_quantize(I_hmmd(ii,jj,1),I_hmmd(ii,jj,2),I_hmmd(ii,jj,3));
       end
   end

   %% 2 Escombrat de la imatge amb la bola 4x4

   %Cada posicio "k" del csd_hist representa el numero de boles 8x8 en las que
   %apareix al menys una vegada el bin "k"
   csd_hist = zeros(1, 256);  % Inicializa histograma

   tamany_bola = 4;

   I_q = I_hmmd_quantized;
   I_files = size(I_q,1);
   I_columnes = size(I_q,2);

   for kk = 1:(I_files - tamany_bola + 1)
       for ll = 1:(I_columnes - tamany_bola + 1)

           % Extraer bloque 8x8 con solapamiento
           block = I_q(kk:kk+tamany_bola-1, ll:ll+tamany_bola-1);

           % Encuentra los bins únicos presentes en el bloque
```

```
            bins_present = unique(block);

            % Incrementa el histograma una vez por bin presente en el bloque
            csd_hist(bins_present + 1) = csd_hist(bins_present + 1) + 1;
        end
    end


    %% 3 Normalitzacio i quantitzacio no lineal

    % normalitzacio
    csd_hist = csd_hist / sum(csd_hist);

    % quantitzacio no lineal
    a = 0.4;
    csd_quantized = round( (log(1 + a * csd_hist) / log(1 + a)) * 255 );
    H_csd_bola4x4(i+1,:) = csd_quantized;
end
toc
disp(['Hora finalizacion procesado pixel a pixel bola 4x4 ', num2str(num_images),' imagenes con
', num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
save(['H_CSD_bola4x4_',num2str(num_images),'imgs_',num2str(hist_bins),'bins_pixelApixel'],"H_csd_
bola4x4")
```

## A1.6 Bucle principal i funcions modificades de la configuració “128”

### Bucle principal `Prog2_CSD_Feature_Extraction_128.m`

```
clear all
% Define Image Database directory and images filename prefix
ImDB_path = [pwd,'\UKentuckyDatabase\'];
ImTest_path = [pwd,'\img\'];
ImDB_name_prefix = 'ukbench';
% Parameters for feature extraction
hist_bins = 128;
num_images = 2000;
H_csd_128 = zeros(num_images,hist_bins);
tic
disp(['Hora inicio procesado pixel a pixel ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
%% Inici bucle principal
for i = 0:(num_images-1)
   filename = [ImDB_path, ImDB_name_prefix, sprintf('%05d.jpg',i)];
   %% 1.1 Conversió a l'espai de color HMMD
   mode = 'HDS'; %mode pot ser HMM o HDS

   % conversio pixel a pixel (amb "prog2_rgb2hmmd.m")
   I_hmmd = prog2_rgb2hmmd(imread(filename),mode);

   %% 1.2 Quantitzacio no uniforme de la imatge hmmd

   % Cuantizacion por pixel (usando "prog2_hmmd_quantize.m")
   I_files = size(I_hmmd,1);
   I_columnes = size(I_hmmd,2);
   I_hmmd_quantized = zeros(I_files,I_columnes);
   for ii = 1:(I_files)
       for jj = 1:I_columnes

I_hmmd_quantized(ii,jj)=prog2_hmmd_quantize_128(I_hmmd(ii,jj,1),I_hmmd(ii,jj,2),I_hmmd(ii,jj,3));
       end
   end
   %% 2 Escombrat de la imatge amb la bola 8x8

   %Cada posicio "k" del csd_hist representa el numero de boles 8x8 en las que
   %apareix al menys una vegada el bin "k"
   csd_hist = zeros(1, hist_bins);  % Inicializa histograma

   tamany_bola = 8;

   I_q = I_hmmd_quantized;
   I_files = size(I_q,1);
   I_columnes = size(I_q,2);

   for kk = 1:(I_files - tamany_bola + 1)
       for ll = 1:(I_columnes - tamany_bola + 1)

           % Extraer bloque 8x8 con solapamiento
           block = I_q(kk:kk+tamany_bola-1, ll:ll+tamany_bola-1);

           % Encuentra los bins únicos presentes en el bloque
```

```
            bins_present = unique(block);

            % Incrementa el histograma una vez por bin presente en el bloque
            csd_hist(bins_present + 1) = csd_hist(bins_present + 1) + 1;
        end
    end


    %% 3 Normalitzacio i quantitzacio no lineal

    % normalitzacio
    csd_hist = csd_hist / sum(csd_hist);

    % quantitzacio no lineal
    a = 0.4;
    csd_quantized = round( (log(1 + a * csd_hist) / log(1 + a)) * 255 );
    H_csd_128(i+1,:) = csd_quantized;
end
toc
disp(['Hora finalizacion procesado pixel a pixel ', num2str(num_images),' imagenes con ',
num2str(hist_bins), ' bins: '   datestr(now, 'dd/mm/yy-HH:MM')]);
save(['H_CSD_',num2str(num_images),'imgs_',num2str(hist_bins),'bins_pixelApixel'],"H_csd_128")
```

## Funció `prog2_hmmd_quantize_128.m`

```
function bin = prog2_hmmd_quantize_128(H, D, S)
% Donat un pixel HMMD (H,D,S) (normalitzat, de 0 a 1) aquesta funció
% retorna un numero entre 0 i 255, indicant el bin del histograma al qual
% pertany.
diff_thresholds = [6, 20, 60, 110] / 255;
%         Hue  Sum
levels = [ 1, 16;  % Subespai 0 - 1*16=16 + 0 = 16 | 1r offset
           4,  4;  % Subespai 1 - 4*4=16 + 16 = 32 | 2n offset
           8,  4;  % Subespai 2 - 8*4=32 + 32 = 64 | 3r offset
           8,  4;  % Subespai 3 - 8*4=32 + 64 = 96 | 4t offset
           8,  4]; % Subespai 4 - 8*4=32 + 96 = 128
% Offset per a cada zona de l'histograma
zone_offsets = [0, 16, 32, 64, 96];  % acumulacio de bins per subespai
% Deteccio de subespai
zone = find(D < diff_thresholds, 1);
if isempty(zone)
    zone = 5;
end
hue_levels = levels(zone,1);
sum_levels = levels(zone,2);
% Evitar desbordaments per valors = 1.0
epsilon = 1e-10;
H = min(H, 1 - epsilon);
S = min(S, 1 - epsilon);
% Quantització
h_q = floor(H * hue_levels);
s_q = floor(S * sum_levels);
% Índex global del bin
bin = zone_offsets(zone) + s_q * hue_levels + h_q;
end
```

## A1.7 Script gràfic comparació de resultats

Aquest script `Prog2_Comparativa.m` genera el gràfic que compara l'fscore de les diferents configuracions:

```
clear all
%% Cargamos precision i recalls i calculamos F-score
% 256_base
load("precision_256_base.mat","precision_avg")
precision_256_base = precision_avg;
load("recall_256_base.mat","recall_avg")
recall_256_base = recall_avg;
f_score_256_base = max((2.*precision_256_base.*recall_256_base)./(precision_256_base+recall_256_base));
% 256_delmat
load("precision_256_delmat.mat","precision_avg")
precision_256_delmat = precision_avg;
load("recall_256_delmat.mat","recall_avg")
recall_256_delmat = recall_avg;
f_score_256_delmat =
max((2.*precision_256_delmat.*recall_256_delmat)./(precision_256_delmat+recall_256_delmat));
% 256_opt
load("precision_256_opt.mat","precision_avg")
precision_256_opt = precision_avg;
load("recall_256_opt.mat","recall_avg")
recall_256_opt = recall_avg;
f_score_256_opt = max((2.*precision_256_opt.*recall_256_opt)./(precision_256_opt+recall_256_opt));
% 4x4
load("precision_256_4x4.mat","precision_avg")
precision_256_4x4 = precision_avg;
load("recall_256_4x4.mat","recall_avg")
recall_256_4x4 = recall_avg;
f_score_256_4x4 = max((2.*precision_256_4x4.*recall_256_4x4)./(precision_256_4x4+recall_256_4x4));
% 128
load("precision_128.mat","precision_avg")
precision_128 = precision_avg;
load("recall_128.mat","recall_avg")
recall_128 = recall_avg;
f_score_128 = max((2.*precision_128.*recall_128)./(precision_128+recall_128));
%% gráfico del precision recall
perfect_precision = [1,1,1,1,4/5,4/6,4/7,4/8,4/9,4/10];
perfect_recall = [0.25,0.5,0.75,1,1,1,1,1,1,1];
% contour per les isolines
x = linspace(0,1);
y = linspace(0,1);
[X,Y] = meshgrid(x,y); % contour nomes accepta matrius
Z = (2.*X.*Y)./(X+Y); % formula del f-score per a que calculi les isolinies
contour(X,Y,Z)
hold on
plot(perfect_recall,perfect_precision,'k-diamond','LineWidth',2)
grid on
hold on
xlabel('Recall');
ylabel('Precision');
title('Grafic Precision-Recall | Distancia: Hellinger')
plot(recall_256_base,precision_256_base,'-+')
plot(recall_256_opt,precision_256_opt,'--o')
plot(recall_256_delmat,precision_256_delmat,'-*')
plot(recall_256_4x4,precision_256_4x4,':^')
plot(recall_128,precision_128,'-.v')
legend('isoFScore','Optimal (F=1)',strcat('256 base (F=',num2str(f_score_256_base),')'),
   strcat('256 opt (F=',num2str(f_score_256_opt),')'),strcat('256 delmat
(F=',num2str(f_score_256_delmat),')'),
   strcat('256 4x4 (F=',num2str(f_score_256_4x4),')'),strcat('128
(F=',num2str(f_score_128),')'),'Location','southwest')
```

# Annex 2: Fitxer `output4.txt`

Aquest és el fitxer de resultats que es va entregar la darrera sessió de laboratori dedicada a aquest projecte. Correspon a la configuració “256_base”, amb un F-score de 0,8625

```
Retrieved list for query image ukbench01701.jpg
ukbench01701.jpg
ukbench01703.jpg
ukbench01702.jpg
ukbench01700.jpg
ukbench01809.jpg
ukbench01808.jpg
ukbench01810.jpg
ukbench01811.jpg
ukbench01743.jpg
ukbench01740.jpg

Retrieved list for query image ukbench00926.jpg
ukbench00926.jpg
ukbench00924.jpg
ukbench00927.jpg
ukbench00917.jpg
ukbench00925.jpg
ukbench00918.jpg
ukbench00973.jpg
ukbench00916.jpg
ukbench00919.jpg
ukbench00972.jpg

Retrieved list for query image ukbench01883.jpg
ukbench01883.jpg
ukbench01882.jpg
ukbench00883.jpg
ukbench00881.jpg
ukbench01880.jpg
ukbench00880.jpg
ukbench00882.jpg
ukbench01881.jpg
ukbench01957.jpg
ukbench01914.jpg

Retrieved list for query image ukbench00116.jpg
ukbench00116.jpg
ukbench00118.jpg
ukbench00117.jpg
ukbench00119.jpg
ukbench00100.jpg
ukbench00102.jpg
ukbench00103.jpg
ukbench00082.jpg
ukbench00101.jpg
ukbench00519.jpg

Retrieved list for query image ukbench00213.jpg
ukbench00213.jpg
ukbench00215.jpg
ukbench00214.jpg
ukbench01177.jpg
ukbench01176.jpg
ukbench01148.jpg
ukbench01178.jpg
ukbench01437.jpg
ukbench01438.jpg
ukbench00661.jpg

Retrieved list for query image ukbench00771.jpg
ukbench00771.jpg
ukbench00770.jpg
ukbench00769.jpg
ukbench00768.jpg
ukbench00146.jpg
ukbench00136.jpg
ukbench00860.jpg
ukbench00147.jpg
ukbench00121.jpg
ukbench00122.jpg

Retrieved list for query image ukbench01776.jpg
ukbench01776.jpg
ukbench01777.jpg
ukbench01778.jpg
ukbench00944.jpg
ukbench01900.jpg
ukbench00211.jpg
ukbench01749.jpg
ukbench01779.jpg
ukbench00208.jpg
ukbench01913.jpg

Retrieved list for query image ukbench01507.jpg
ukbench01507.jpg
ukbench01504.jpg
ukbench01505.jpg
ukbench01690.jpg
ukbench01657.jpg
ukbench01506.jpg
ukbench01656.jpg
ukbench01659.jpg
ukbench01689.jpg
ukbench01497.jpg

Retrieved list for query image ukbench00440.jpg
ukbench00440.jpg
ukbench00442.jpg
ukbench00443.jpg
ukbench00441.jpg
ukbench01135.jpg
ukbench00955.jpg
ukbench00393.jpg
ukbench00392.jpg
ukbench00952.jpg
ukbench00953.jpg

Retrieved list for query image ukbench00903.jpg
ukbench00903.jpg
ukbench00901.jpg
ukbench00900.jpg
ukbench00902.jpg
ukbench00815.jpg
ukbench00856.jpg
ukbench00812.jpg
ukbench00813.jpg
ukbench00843.jpg
ukbench00814.jpg
```

```
Retrieved list for query image ukbench00693.jpg
ukbench00693.jpg
ukbench00695.jpg
ukbench00692.jpg
ukbench00595.jpg
ukbench00752.jpg
ukbench00847.jpg
ukbench00331.jpg
ukbench01929.jpg
ukbench00846.jpg
ukbench00620.jpg

Retrieved list for query image ukbench00379.jpg
ukbench00379.jpg
ukbench00378.jpg
ukbench00376.jpg
ukbench00377.jpg
ukbench00395.jpg
ukbench00393.jpg
ukbench00165.jpg
ukbench00443.jpg
ukbench00167.jpg
ukbench00383.jpg

Retrieved list for query image ukbench00466.jpg
ukbench00466.jpg
ukbench00464.jpg
ukbench00465.jpg
ukbench00961.jpg
ukbench00962.jpg
ukbench00481.jpg
ukbench00480.jpg
ukbench00483.jpg
ukbench00960.jpg
ukbench00450.jpg

Retrieved list for query image ukbench00600.jpg
ukbench00600.jpg
ukbench00601.jpg
ukbench00609.jpg
ukbench00610.jpg
ukbench00685.jpg
ukbench00603.jpg
ukbench00608.jpg
ukbench00602.jpg
ukbench00674.jpg
ukbench00673.jpg

Retrieved list for query image ukbench00458.jpg
ukbench00458.jpg
ukbench00456.jpg
ukbench00457.jpg
ukbench00459.jpg
ukbench00010.jpg
ukbench00009.jpg
ukbench00360.jpg
ukbench00065.jpg
ukbench00011.jpg
ukbench00415.jpg
```

```
Retrieved list for query image ukbench01350.jpg
ukbench01350.jpg
ukbench01351.jpg
ukbench01349.jpg
ukbench01348.jpg
ukbench01451.jpg
ukbench01450.jpg
ukbench01448.jpg
ukbench01449.jpg
ukbench01408.jpg
ukbench01215.jpg

Retrieved list for query image ukbench00804.jpg
ukbench00804.jpg
ukbench00806.jpg
ukbench00807.jpg
ukbench00834.jpg
ukbench00854.jpg
ukbench00805.jpg
ukbench00833.jpg
ukbench00832.jpg
ukbench00852.jpg
ukbench00835.jpg

Retrieved list for query image ukbench00601.jpg
ukbench00601.jpg
ukbench00602.jpg
ukbench00600.jpg
ukbench00603.jpg
ukbench00685.jpg
ukbench00609.jpg
ukbench00674.jpg
ukbench00673.jpg
ukbench00608.jpg
ukbench00610.jpg

Retrieved list for query image ukbench01998.jpg
ukbench01998.jpg
ukbench01999.jpg
ukbench01997.jpg
ukbench01996.jpg
ukbench00881.jpg
ukbench00882.jpg
ukbench00833.jpg
ukbench00883.jpg
ukbench00832.jpg
ukbench01880.jpg

Retrieved list for query image ukbench00801.jpg
ukbench00801.jpg
ukbench00803.jpg
ukbench00802.jpg
ukbench00800.jpg
ukbench00796.jpg
ukbench00797.jpg
ukbench00784.jpg
ukbench00799.jpg
ukbench00798.jpg
ukbench01973.jpg
```

# Annex 3: Fitxer `output4_256_delmat.txt`

Aquest és el fitxer de resultats que es va generar amb la versió alternativa en la qual s'aplica delmat abans de fer l'escombrat. Correspon a la configuració "256_delmat", amb un F-score de 0,9.

```
Retrieved list for query image ukbench01701.jpg
ukbench01701.jpg
ukbench01703.jpg
ukbench01700.jpg
ukbench01702.jpg
ukbench01809.jpg
ukbench01808.jpg
ukbench01810.jpg
ukbench01743.jpg
ukbench01811.jpg
ukbench00851.jpg

Retrieved list for query image ukbench00926.jpg
ukbench00926.jpg
ukbench00924.jpg
ukbench00927.jpg
ukbench00917.jpg
ukbench00973.jpg
ukbench00972.jpg
ukbench00974.jpg
ukbench00918.jpg
ukbench00916.jpg
ukbench00919.jpg

Retrieved list for query image ukbench01883.jpg
ukbench01883.jpg
ukbench01882.jpg
ukbench01880.jpg
ukbench01881.jpg
ukbench00883.jpg
ukbench00880.jpg
ukbench00881.jpg
ukbench00882.jpg
ukbench00279.jpg
ukbench01999.jpg

Retrieved list for query image ukbench00116.jpg
ukbench00116.jpg
ukbench00117.jpg
ukbench00118.jpg
ukbench00119.jpg
ukbench00100.jpg
ukbench00102.jpg
ukbench00101.jpg
ukbench00103.jpg
ukbench00082.jpg
ukbench00404.jpg

Retrieved list for query image ukbench00213.jpg
ukbench00213.jpg
ukbench00215.jpg
ukbench01148.jpg
ukbench01177.jpg
ukbench01178.jpg
ukbench01149.jpg
ukbench01176.jpg
ukbench01179.jpg
ukbench01150.jpg
```

```
Retrieved list for query image ukbench00771.jpg
ukbench00771.jpg
ukbench00770.jpg
ukbench00769.jpg
ukbench00768.jpg
ukbench00146.jpg
ukbench00136.jpg
ukbench00860.jpg
ukbench00147.jpg
ukbench00259.jpg
ukbench00145.jpg

Retrieved list for query image ukbench01776.jpg
ukbench01776.jpg
ukbench01777.jpg
ukbench01778.jpg
ukbench00944.jpg
ukbench00211.jpg
ukbench01779.jpg
ukbench01900.jpg
ukbench01901.jpg
ukbench00210.jpg
ukbench01892.jpg

Retrieved list for query image ukbench01507.jpg
ukbench01507.jpg
ukbench01504.jpg
ukbench01505.jpg
ukbench01497.jpg
ukbench01690.jpg
ukbench01506.jpg
ukbench01656.jpg
ukbench01657.jpg
ukbench01577.jpg
ukbench01689.jpg

Retrieved list for query image ukbench00440.jpg
ukbench00440.jpg
ukbench00442.jpg
ukbench00443.jpg
ukbench00441.jpg
ukbench01135.jpg
ukbench00955.jpg
ukbench00952.jpg
ukbench00830.jpg
ukbench00828.jpg
ukbench00831.jpg

Retrieved list for query image ukbench00903.jpg
ukbench00903.jpg
ukbench00901.jpg
ukbench00900.jpg
ukbench00902.jpg
ukbench00815.jpg
ukbench00856.jpg
ukbench00812.jpg
ukbench00806.jpg
ukbench00887.jpg
```

```
ukbench01438.jpg

Retrieved list for query image ukbench00693.jpg
ukbench00693.jpg
ukbench00695.jpg
ukbench00692.jpg
ukbench00846.jpg
ukbench00752.jpg
ukbench01939.jpg
ukbench00844.jpg
ukbench00847.jpg
ukbench00153.jpg
ukbench00595.jpg

Retrieved list for query image ukbench00379.jpg
ukbench00379.jpg
ukbench00378.jpg
ukbench00376.jpg
ukbench00377.jpg
ukbench00165.jpg
ukbench00167.jpg
ukbench00395.jpg
ukbench00382.jpg
ukbench00397.jpg
ukbench00381.jpg

Retrieved list for query image ukbench00466.jpg
ukbench00466.jpg
ukbench00464.jpg
ukbench00465.jpg
ukbench00481.jpg
ukbench00961.jpg
ukbench00960.jpg
ukbench00480.jpg
ukbench00483.jpg
ukbench00962.jpg
ukbench00482.jpg

Retrieved list for query image ukbench00600.jpg
ukbench00600.jpg
ukbench00601.jpg
ukbench00602.jpg
ukbench00685.jpg
ukbench00609.jpg
ukbench00603.jpg
ukbench00610.jpg
ukbench00608.jpg
ukbench00674.jpg
ukbench00673.jpg

Retrieved list for query image ukbench00458.jpg
ukbench00458.jpg
ukbench00456.jpg
ukbench00457.jpg
ukbench00459.jpg
ukbench00010.jpg
ukbench00009.jpg
ukbench00360.jpg
ukbench00065.jpg
ukbench00713.jpg
ukbench00415.jpg
```

```
ukbench00814.jpg

Retrieved list for query image ukbench01350.jpg
ukbench01350.jpg
ukbench01351.jpg
ukbench01349.jpg
ukbench01348.jpg
ukbench01450.jpg
ukbench01448.jpg
ukbench01451.jpg
ukbench01449.jpg
ukbench01408.jpg
ukbench01250.jpg

Retrieved list for query image ukbench00804.jpg
ukbench00804.jpg
ukbench00806.jpg
ukbench00807.jpg
ukbench00805.jpg
ukbench00834.jpg
ukbench00833.jpg
ukbench00854.jpg
ukbench00077.jpg
ukbench00843.jpg
ukbench00832.jpg

Retrieved list for query image ukbench00601.jpg
ukbench00601.jpg
ukbench00602.jpg
ukbench00600.jpg
ukbench00603.jpg
ukbench00685.jpg
ukbench00673.jpg
ukbench00674.jpg
ukbench00609.jpg
ukbench00608.jpg
ukbench00610.jpg

Retrieved list for query image ukbench01998.jpg
ukbench01998.jpg
ukbench01999.jpg
ukbench01996.jpg
ukbench01997.jpg
ukbench01876.jpg
ukbench01878.jpg
ukbench00880.jpg
ukbench00883.jpg
ukbench00833.jpg
ukbench00834.jpg

Retrieved list for query image ukbench00801.jpg
ukbench00801.jpg
ukbench00803.jpg
ukbench00802.jpg
ukbench00800.jpg
ukbench00796.jpg
ukbench00797.jpg
ukbench00799.jpg
ukbench00798.jpg
ukbench00784.jpg
ukbench00785.jpg
```

# Annex 4: Treball previ i comprovacions manuals

Script que compara la nostra funció que passa de RGB a HSV amb la que proporciona Matlab:

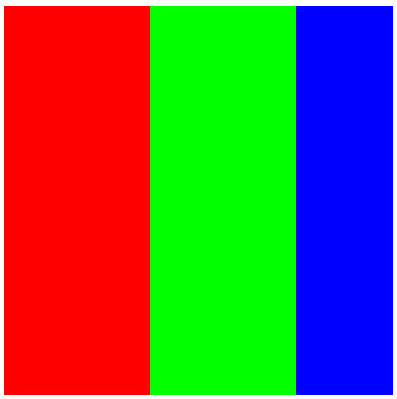
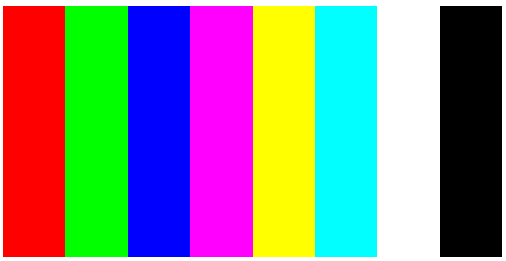
```
clear all
% Define Image Database directory and images filename prefix
ImDB_path = [pwd,'\UKentuckyDatabase\'];
ImTest_path = [pwd,'\img\'];
ImDB_name_prefix = 'ukbench';
% Noms imatges test:
% '8x8_test_rgb_vertical_strip.png'
% '16x8_test_rgb_rgb-barreja_vertical.png'
filename_test = [ImTest_path, '16x8_test_rgb_rgb-barreja_vertical.png'];
%% Implementació de funció prog2_rgb2hsv per veure que estem implementant
%  bé una funció d'aquest tipus, comparant la nostra amb la propia de
%  Matlab. L'error absolut es tan baix que la donem per bona
I_hsv = rgb2hsv(imread(filename));
I_prog2_hsv = prog2_rgb2hsv(imread(filename));
max_abs_error = max(abs(I_hsv(:) - I_prog2_hsv(:)));
disp(['Error absoluto máximo: ', num2str(max_abs_error)]);
if max_abs_error < 1e-10
   disp(['Error indistinguible']);
else
   disp(['Error notable']);
end
```

Funció `prog2_rgb2hsv.m`:

```
function [hsv_image_out] = prog2_rgb2hsv(rgb_image_in)
% Implementacio de la funcio rgb2hsv
% La entrada es una imatge de l'espai de color RGB
% La sortida es la imatge d'entrada convertida a l'espai de color HSV
hsv_image_out = zeros(size(rgb_image_in,1),size(rgb_image_in,2),3);
rgb_image_in = double(rgb_image_in) / 255; % Es normalitza la imatge d'entrada
for i = 1:size(rgb_image_in,1)
   for j = 1:size(rgb_image_in,2)
       % Calcul de Max i Min
       maxx=max(rgb_image_in(i,j,:));
       minn=min(rgb_image_in(i,j,:));
       % Calcul del Hue
       r = rgb_image_in(i,j,1);
       g = rgb_image_in(i,j,2);
       b = rgb_image_in(i,j,3);
       if maxx == minn
           hue = 0;
       elseif (maxx == r) && (g >= b)
           hue = 60*((g-b)/(maxx-minn));
       elseif (maxx == r) && (g < b)
           hue = 360 + 60*((g-b)/(maxx-minn));
       elseif g == maxx
           hue = 60*(2.0+ ((b-r)/(maxx-minn)));
       else
           hue = 60*(4.0+ ((r-g)/(maxx-minn)));
       end
       hue = hue / 360; % Es normalitza el valor de Hue
       % Calcul del Saturation
       if maxx == 0
           saturation = 0;
       else
           saturation = (maxx-minn)/maxx;
       end
```

```
        value=maxx;
        hsv_image_out(i,j,1)=hue;
        hsv_image_out(i,j,2)=saturation;
        hsv_image_out(i,j,3)=value;
    end
end
end
```

Imatges test:

| 8x8_test_rgb_vertical_strip.png | 16x8_test_rgb_rgb-barreja_vertical.png |
| --- | --- |
|  |  |

Les imatges son del tamany que indica el seu nom, però es veuen “borroses” per l’escalat que fa el Word al fer-les més gran (per a que es puguin veure).

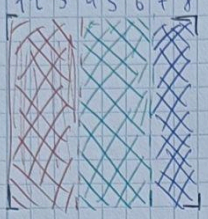
Comprovacions manuals de ambdues imatges:

PIV - proj 2 - CSD

## (1) rgb → HMMD

* Imatge test "8x8_test_rgb_vertical_stripping.png"

columnes 1 2 3 4 5 6 7 8



(255, 0, 0) → (1, 0, 0) (R, G, B)

(0, 255, 0) → (0, 1, 0)

(0, 0, 255) → (0, 0, 1)

+ HMMD → cada pixel (H, D, S) → Hue, Diff, Sum

* Per cada pixel RGB:

Max = max(R, G, B)

Min = min(R, G, B)

Diff = Max - Min

Sum = $\frac{Max + Min}{2}$

$$Hue = \begin{cases} 0 & \text{si } Max == Min \\ 60 \cdot \frac{G-B}{Max-Min} & \text{si } Max == R \text{ i } G \geq B \\ 360 + 60 \cdot \frac{G-B}{Max-Min} & \text{si } Max == R \text{ i } G < B \\ 60 \cdot \left(2,0 + \frac{B-R}{Max-Min}\right) & \text{si } G == Max \\ 60 \cdot \left(4,0 + \frac{R-G}{Max-Min}\right) & \text{altre} \end{cases}$$

Per un pixel vermell (1, 0, 0) → [(0, 1, 0'5)]

Max = 1 Diff = 1 Hue = $60 \cdot \frac{G-B}{Max-Min} = 60 \cdot \frac{0-0}{1-0} = 0$

Min = 0 Sum = 0,5

Per un pixel verd (0, 1, 0) → [(0'3̂, 1, 0'5)]

Max = 1 Diff = 1 Hue = $60 \cdot \left(2,0 + \frac{B-R}{Max-Min}\right) = 60\left(2,0 + \frac{0-0}{1-0}\right) = 120$

Min = 0 Sum = 0,5 120/360 = 0,3̂

Per un pixel blau (0, 0, 1) → [(0'6̂, 1, 0'5)]

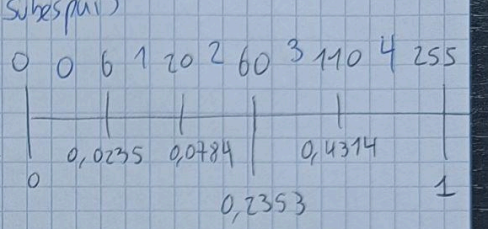
Max = 1 Diff = 1 Hue = $60\left(4,0 + \frac{R-G}{Max-Min}\right) = 60\left(4,0 + \frac{0-0}{1-0}\right) = 240$

Min = 0 Sum = 0,5 240/360 = 0,6̂

La imatge RGB 8x8 queda en HMMD com

| Columnes | RGB | HMMD (Hue, Diff, Sum) |
|---|---|---|
| 1,2,3 | (1,0,0) | (0, 1, 0'5) |
| 4,5,6 | (0,1,0) | (0'$\hat{3}$, 1, 0'5) |
| 7,8 | (0,0,1) | (0'$\hat{6}$, 1, 0'5) |

## (2) Quantització no uniforme




* L'eix "Diff" es talla en 5 sub-intervals
  ↳ Normalitzat

* Pels 3 colors Diff = 1
  ↳ "caven" al subespai 4

| 256 bins | | |
|---|---|---|
| subespai | Hue | Sum |
| 0 | 1 | 32 |
| 1 | 4 | 8 |
| 2,3,4 | 16 | 4 |

⇒ Hue = 16 Sum = 4

Hue quantitzat: R floor $(0 \cdot 16) = 0$

G floor $(0{,}\hat{3} \cdot 16) = \text{floor}(5{,}\hat{3}) = 5$

B floor $(0{,}\hat{6} \cdot 16) = \text{floor}(10{,}\hat{6}) = 10$

Sum quantitzat: RGB floor $(0{,}5 \cdot 4) = 2$

Càlcul del bin en subespai 4 (offset = 192)

bin = offset + sum_quantitzat · hue_level + hue_quantitzat

R: bin = 192 + 2·16 + 0 = 224

G: bin = 192 + 2·16 + 5 = 229

B: bin = 192 + 2·16 + 10 = 234

Resultat:

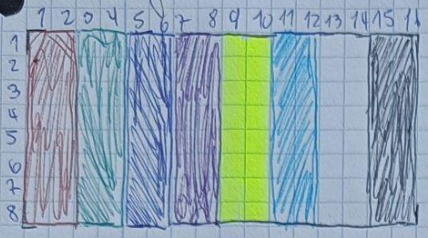
| Columnes imatge | RGB | HMMD | Quantitzat |
|---|---|---|---|
| 1,2,3 | (1,0,0) | (0, 1, 0'5) | 224 |
| 4,5,6 | (0,1,0) | (0'$\hat{3}$, 1, 0'5) | 229 |
| 7,8 | (0,0,1) | (0'$\hat{6}$, 1, 0'5) | 234 |

Si se li fa l'histograma (SD), només cap 1 únic cop la bola 8x8

PIV - prog2 - CSD

## ① RGB → HMMD

* Imatge test "16x8_test_rgb_rgb-barres_vertical.png"



RGB

(1,0,0) (1,1,0)
(0,1,0) (0,1,1)
(0,0,1) (1,1,1)
(1,0,1) (0,0,0)

→ pasar colors a HMMD

- Vermell $(1,0,0)_{RGB} \rightarrow (0, 1, 0'5)_{HMMD}$
- Verd $(0,1,0)_{RGB} \rightarrow (0'\hat{3}, 1, 0'5)_{HMMD}$
- Blau $(0,1,0)_{RGB} \rightarrow (0'\hat{6}, 1, 0'5)_{HMD}$
- Lila $(1,0,1)_{RGB} \rightarrow (0'8\hat{3}, 1, 0'5)_{HMMD}$

  Max = 1 Diff = 1 $Hue = 360 + 60 \cdot \frac{0-1}{1-0} = 300 \rightarrow 300/360 = 0,8\hat{3}$
  Min = 0 Sum = 0'5
- Groc $(1,1,0)_{RGB} \rightarrow (0'1\hat{6}, 1, 0'5)_{HMMD}$

  Max = 1 Diff = 1 $Hue = 60 \cdot \frac{1-0}{1-0} = 60$ $60/360 = 0'1\hat{6}$
  Min = 0 Sum = 0'5
- Cyan $(0,1,1)_{RGB} \rightarrow (0'5, 1, 0'5)_{HMMD}$

  Max = 1 Diff = 1 $Hue = 60 \cdot \left(2 + \frac{1-0}{1-0}\right) = 180$ $180/360 = 0'5$
  Min = 0 Sum = 0'5
- Blanc $(1,1,1)_{RGB} \rightarrow (0, 0, 1)_{HMMD}$

  Max = 1 Diff = 0 Hue = 0
  Min = 1 Sum = 1
- Negre $(0,0,0)_{RGB} \rightarrow (0, 0, 0)_{HMMD}$

  Max = 0 Diff = 0 Hue = 0
  Min = 0 Sum = 0

| RUB | HMMD |
|---|---|
| (1,0,0) | (0, 1, 0'5) |
| (0,1,0) | (0'3̂, 1, 0'5) |
| (0,0,1) | (0'6̂, 1, 0'5) |
| (1,0,1) | (0'83̂, 1, 0'5) |
| (1,1,0) | (0'16̂, 1, 0'5) |
| (0,1,1) | (0'5, 1, 0'5) |
| (1,1,1) | (0,0,1) |
| (0,0,0) | (0,0,0) |

| Subespai | 256 bins Hue | Sum |
|---|---|---|
| 0 | 1 | 32 |
| 1 | 4 | 8 |
| 2,3,4 | 16 | 4 |

(2) Quantització: Diff (Vermell, Verd, Blau, Lila, Groc, Cyan) = 1
↳ Subespai 4 ⇒ offset = 192
↳ Hue = 16 Sum = 4

Diff (Blanc, Negre) = 0
↳ Subespai 0 ⇒ offset = 0, Hue = 1 Sum = 32

Vermell: Hue = floor(0·16) = 0
Sum = floor(0'5·4) = 2

Verd: Hue = floor(0'3̂·16) = 5
Sum = floor(0'5·4) = 2

Blau: Hue = floor(0'6̂·16) = 10
Sum = floor(0'5·4) = 2

Lila: Hue = floor(0'83̂·16) = 13
Sum = floor(0'5·4) = 2

Groc: Hue = floor(0'16̂·16) = 2
Sum = floor(0'5·4) = 2

Cyan: Hue = floor(0'5·16) = 8
Sum = floor(0'5·4) = 2

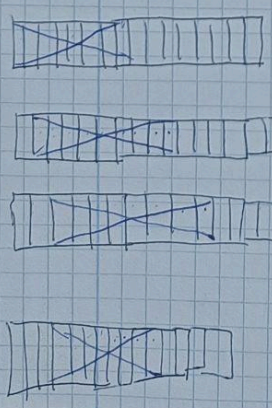
Blanc: Hue = floor(0·1) = 0
Sum = floor(1·32) = 31

Negre: Hue = floor(0·1) = 0
Sum = floor(0·32) = 0

bin = offset + sum_quantitzat · hue_level + hue_quantitzat

Vermell: 192 + 2·16 + 0 = 224
Verd: 192 + 2·16 + 5 = 229
Blau: 192 + 2·16 + 10 = 234
Lila: 192 + 2·16 + 13 = 237

Groc: 192 + 2·16 + 2 = 226
Cyan: 192 + 2·16 + 8 = 232
Blanc: 0 + 32·1 + 0 = 31
Negre: 0 + 0·1 + 0 = 0



0: 224:1 226:0 / 229:1 232:0 / 234:1 31:0 / 237:1 0:0
1: 224:2 226:1 / 229:2 232:0 / 234:2 31:0 / 237:2 0:0
2: 224:2 226:2 / 229:3 232:0 / 234:3 31:0 / 237:3 0:0
3: 224:2 226:3 / 229:4 232:1 / 234:4 31:0 / 237:4 0:0
4: 224:2 226:4 / 229:4 232:2 / 234:5 31:0 / 237:5 0:0
5: 224:2 226:5 / 229:4 232:3 / 234:6 31:1 / 237:6 0:0
6: 224:2 226:6 / 229:4 232:4 / 234:6 31:2 / 237:7 0:0
7: 224:2 226:7 / 229:4 232:5 / 234:6 31:3 / 237:8 0:1
8: 224:2 226:8 / 229:4 232:6 / 234:6 31:4 / 237:8 0:2

Gsd_hist final
resta bins 0