MEMÒRIA

Programació 2: SCD

Júlia Blanco
Laura Rayón
G21

ÍNDEX

Descripció Global del Sistema

El cercador d'imatges similars d'aquesta segona part, està basat en el Scalable Color Descriptor de l'estàndard MPEG-7. Per poder implementar-lo, hem fet modificacions al Prog1, obtenint la següent estructura de programa:

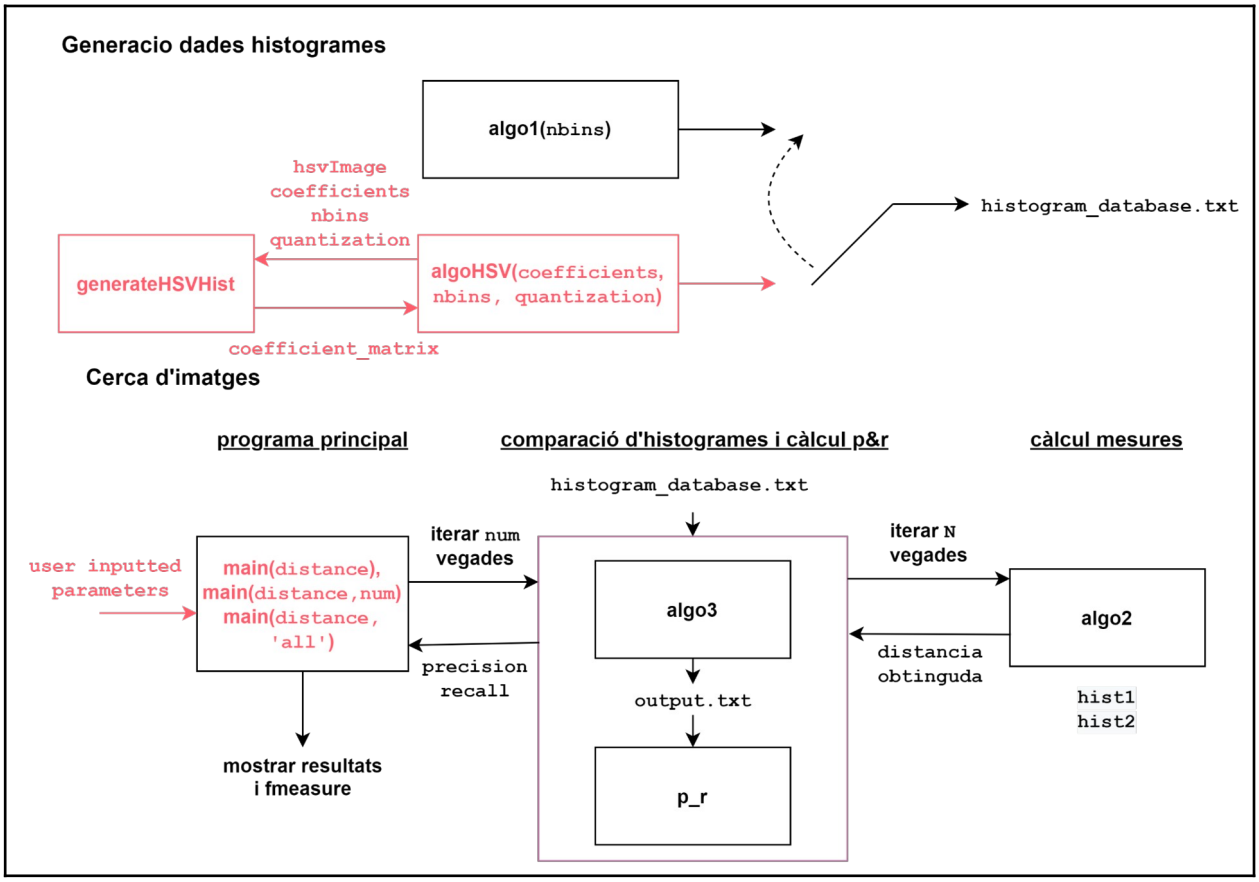
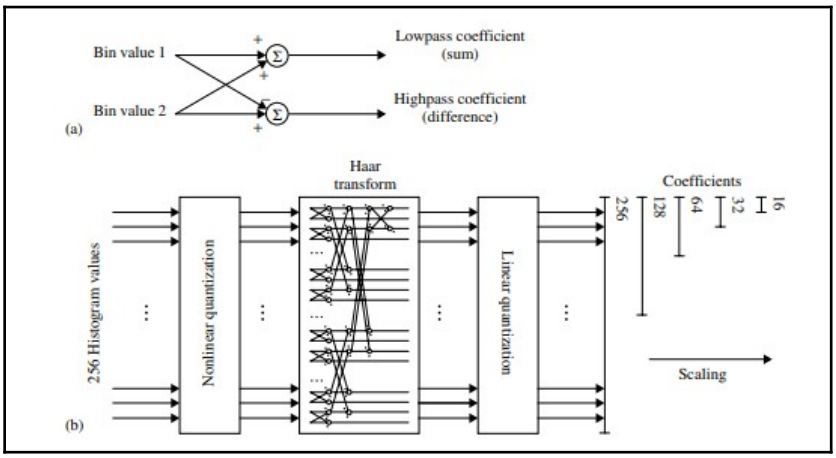


Diagrama 1: Estructura del prog2: *En vermell les modificacions respecte prog1.*

FUNCIONAMENT DE PROG 2

1. Generació dels histogrames de les imatges

En prog2, els valors que serviran per caracteritzar les imatges i poder-les comparar entre elles són els coeficients del Scalable Color Descriptor. L'obtenció d'aquests coeficients s'ha implementat en la funció **generateHSVHist** seguint l'estructura indicada a [1] i que es mostra en la imatge 1.



Imatge 1: (a)Unitat bàsica de la transformada de Haar i (b) diagrama esquemàtic de la generació del SCD. Imatge extreta de [1]

1.1. Implementació de generateHSVHist

1.1.1. Generació histograma HSV

El primer pas és generar l'histograma de la imatge a processar en l'espai de color HSV. Els nivells de H, S i V vénen determinats pel número de bins amb que l'usuari ha indicat que vol generar l'histograma, seguint el criteri següent:

Number of coefficient ss	Number of bins H	Number of bins S	Number of bins V
16	4	2	2
32	8	2	2
64	8	2	4
128	8	4	4
256	16	4	4

Imatge 2: Nivells utilitzats per les particions de l'espai HSV en funció del nombre de coeficients en l'SCD

```
% select bit amount accordingly
switch nbins
    case 256
        h_levels = 16;                    % # of hue levels
        s_levels = 4;                     % # of saturation levels
        v_levels = 4;                     % # of value levels
    case 128
        h_levels = 8;                     % # of hue levels
        s_levels = 4;                     % # of saturation levels
        v_levels = 4;                     % # of value levels
    case 64
        h_levels = 8;                     % # of hue levels
        s_levels = 2;                     % # of saturation levels
        v_levels = 4;                     % # of value levels
    case 32
```

```
        h_levels = 8;                                   % # of hue levels
        s_levels = 2;                                   % # of saturation levels
        v_levels = 2;                                   % # of value levels
    case 16
        h_levels = 4;                                   % # of hue levels
        s_levels = 2;                                   % # of saturation levels
        v_levels = 2;                                   % # of value levels
end
h_bits = log2(h_levels);                                % # of hue bits
s_bits = log2(s_levels);                                % # of saturation bits
v_bits = log2(v_levels);                                % # of value bits
total_bits = h_bits+s_bits+v_bits;                      % total bits
```

Un cop escollits els pesos, cada píxel es codifica a l'histograma de la següent manera:

```
%hsvImage has values that range from 0 to 1. We must adapt
%their dynamic range to a value between 0 ... h/s/v_levels-1
hBin = ceil(hsvImage(row, col, 1) * (h_levels-1));  % adapt hue range
sBin = ceil(hsvImage(row, col, 2) * (s_levels-1));  % adapt saturation range
vBin = ceil(hsvImage(row, col, 3) * (v_levels-1));  % adapt value range

% each pixel value is a value that contains information about
% the three histogram bins (h,s,v):
% The h_bits most significant bits are the hue
% The s_bits following bits are the saturation
% The v_bits least significant bits are the value

pixel = 2^(total_bits - h_bits)*hBin+2^(total_bits - h_bits - s_bits)*sBin + vBin;

% Add 1 to the value of the pixel
% since matlab indexes start at 1 and our values range from 0 to level-1,
% a unit is added to the array index to make up for that
hsvHist(pixel+1) = hsvHist(pixel+1)+1;
```

1.1.2. Normalització histograma

El segon pas és normalitzar l'histograma a partir del nombre total de píxels de la imatge. D'aquesta manera els bins prenen valors entre 0 i 1, essent més habituals els valors propers a 0. És per això que en el pas següent s'ofereix la possibilitat d'una quantificació no lineal.

```
% Normalize histogram values
hsvHist = hsvHist/total_pixels; %Returns values between 0 and 1
```

1.1.3. Quantificació lineal o no lineal (a elecció)

Segons els paràmetres escollits per l'usuari, la quantificació de l'histograma es fa de manera lineal o no lineal, per encabir en el nombre de bits especificat a la variable **bits_per_hist_level**. S'ha escollit com a quantificació no lineal la quantificació logarítmica per poder donar més pes a valors petits, ja que això és el que se'ns indica com a necessari a [1].

```
switch quantization
    case 'linear'
        % Linear quantization of coefficients
        qHist = round(hsvHist/max(hsvHist)*(pow2(bits_per_hist_level)-1));
    case 'log'
        % Logarithmic quantization
        hsvHist_log = 10*log10(hsvHist+10^-5);
        % obtain max value of histogram and quantize linearly
        % due to the logarithm and our normalised values, hsvHist_log
        % contains only <0 values, which makes the linear quantization is a bit tricky
```

```
        % (the lowest value is sustracted, which will set it to 0 and
        % increase the 'less negative' ones accordingly ->(hsvHist_log-min(hsvHist_log))
        qHist = round((hsvHist_log - min(hsvHist_log))/max(hsvHist_log -
min(hsvHist_log))*(pow2(bits_per_hist_level) - 1));
end
```

1.1.4. Obtenció coeficients de Haar

El nombre de coeficients de Haar que s'utilitzaràn per fer la comparativa entre imatges és escollible per l'usuari. En funció d'aquest nombre, s'ajuntaran més o menys bins (coeficients pas baix de Haar) segons la pauta d'obtenció de coeficients especificada a [1].

```
index = 1;  % index to fill the resulting array
last_j = nbins-nbins/coefficients-1;
iter = nbins/coefficients; % amount of values that have to be
%accumulated (eg: for 128 coefficients, two sums will be accumulated in each
%bin -> 1+2, 3+4, ...127+128)

%iterates through the entire quantized histogram with iter skips
for j=1:iter:last_j
    %performs the local sums
    for count=1:iter
        haarHist(index)= haarHist(index)+qHist(count+j);    %store values into
haarHist(index)
    end
    index = index+1;    %gets increased every iter times
end
```

1.1.5. Quantificació lineal

Finalment, els coeficients obtinguts es quantitzen per adaptar el seu valor al nombre de bits indicats per la variable **bits_per_hist_level**, és a dir, el mateix valor que en la quantificació de l'histograma normalitzat.

```
% linear quantization of the coefficients
aprox_coef = round(haarHist/max(haarHist)*(pow2(bits_per_hist_level)-1));
```

1.2. Implementació de algoHSV

La funció generateHSVHist s'encarrega del càlcul dels coeficients del SCD per una sola imatge. La funció d'algoHSV és fer córrer generateHSVHist per totes les imatges de la base de dades, i omplir el fitxer **histogram_database.txt** amb els coeficients de cadascuna d'elles. El format d' **histogram_database.txt** és que els coeficients de les imatges es guarden per files, de forma que cada fila correspon a una imatge diferent.

```
%% READ IMAGES AND WRITE THEM INTO THE OUTPUT FILE
fileID = fopen('../histogram_database.txt','w');
for id = 0:N-1
    %obtain image name and retrieve its file
    num = sprintf('%05d', id);
    nameim = strcat('ukbench', num , '.jpg');
    img = imread(nameim);   %read image
    hsvImage = rgb2hsv(img); %convert values RGB->HSV

    %generate coefficients for every image
    [quantized_coefficients] = generateHSVhist(hsvImage, coefficients, nbins,
quantization);

    %print coefficients into histogram_database.txt
    fprintf(fileID,'%d ', quantized_coefficients);
    fprintf(fileID,'\n');
```

```
end
```

2. Cerca d'imatges

La implementació de la cerca d'imatges és molt similar a la proposada en el prog1, la diferència es troba en que en prog2, la funció principal main permet a l'usuari escollir gran part dels paràmetres que s'empraran en la generació dels coeficients de les imatges.

2.1. Main

L'objectiu de la funció main és cercar les imatges que siguin similars a les donades, juntament amb calcular i graficar les corbes de precision & recall.

Per aquesta tasca, la funció ofereix a l'usuari escollir fer la comparació entre imatges amb 3 distàncies diferents (que l'usuari especifica a la variable **distance_method**), i també fer 3 tipus de cerca diferents.

2.2. Altres detalls de la implementació

El càlcul i obtenció de candidats s'ha mantingut respecte al que vam fer en prog1:

- **Distància utilitzada (distance_method):** A elecció de l'usuari com a paràmetre d'entrada (MAE, MSE, 'RMAE'), es calcula per parells d'imatges fent ús de la funció **algo2**.
- **Criteri de decisió:** S'agafen els 10 candidats amb distància mínima. La funció **algo3** és l'encarregada de buscar i escriure els candidats en un fitxer de text de sortida.

Resultats[1]

Les variables amb les quals hem treballat a l'hora d'implementar el nostre prog2, juntament amb els valors implementats han estat:

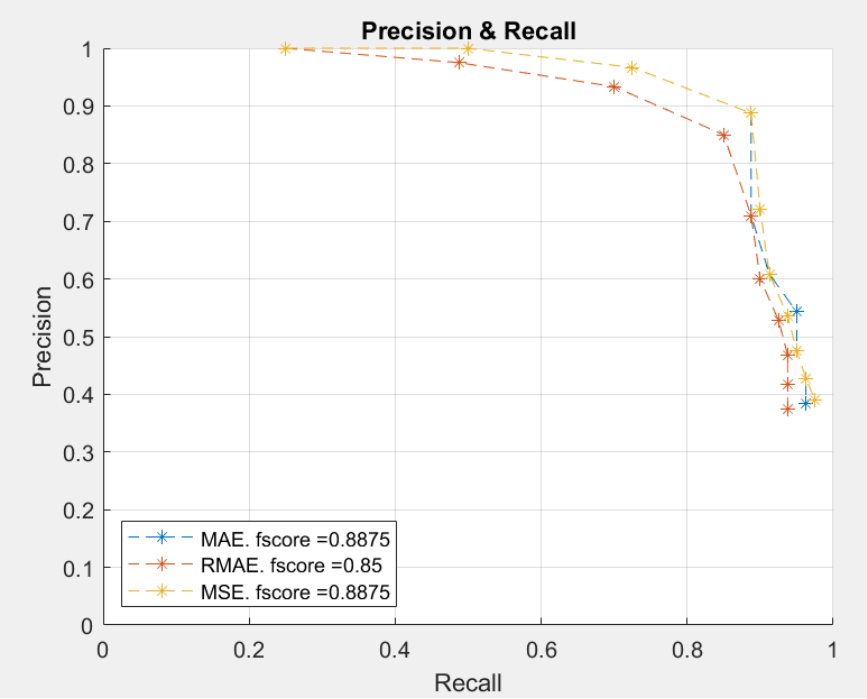
- Distància de comparació: mae, mse, 'rmae'.
- # bins de l'histograma HSV: 256, 128, 64, 32, 16
- Tipus de quantificació de l'histograma HSV normalitzat: lineal, logarítmica
- # bits per codificar la quantificació: 8
- # coeficients Haar usats per definir una imatge: 256, 128, 64, 32, 16

Corbes precisió i recall

1. Cas sense compressió

Primerament, vam fer una prova del funcionament del nostre selector de candidats sense fer cap compressió: utilitzant un histograma de 256 bins i obtenint 256 coeficients. Això ens garanteix pèrdua d'informació mínima i ens pot ajudar a veure quina és la distància que dóna millors resultats.

Amb els resultats veiem que tant MAE com MSE coincideixen en el seu fscore, i que el MSE proporciona uns temps de processat millors.



Gràfica 1: *Precision-Recall obtinguts fent servir histogrames de 256 bins i 256 coeficients (sense compressió) i quantificació logarítmica.*

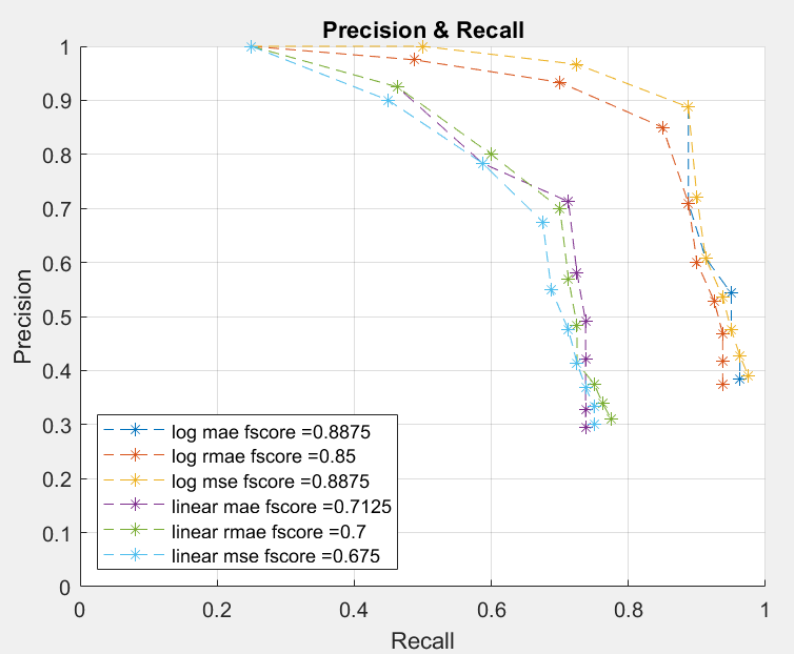
Taula 1: Fscore i temps/imatge de cada distància

Característiques		Distància	Temps total (s)	Temps/imatge (s)	Fscore
#bins HSV histogram	256	MAE	3,5115	0,175575	0,8875
Quantificació	Logarítmica	'RMAE'	3,224	0,1612	0,85
# Haar coefficients	256	MSE	3,0296	0,15148	0,8875

1 obtinguts a partir d'input.txt

2. Comprovació dels efectes de la transformació logarítmica en quantificar

A [1] se'ns indicava que en el primer pas de compressió del SCD donava millors resultats si es feia una compressió no-lineal que donés més pes a valors baixos que una compressió lineal. Per comprovar aquest fet, vam realitzar el mateix experiment que abans, però ara provant també a fer la compressió lineal.

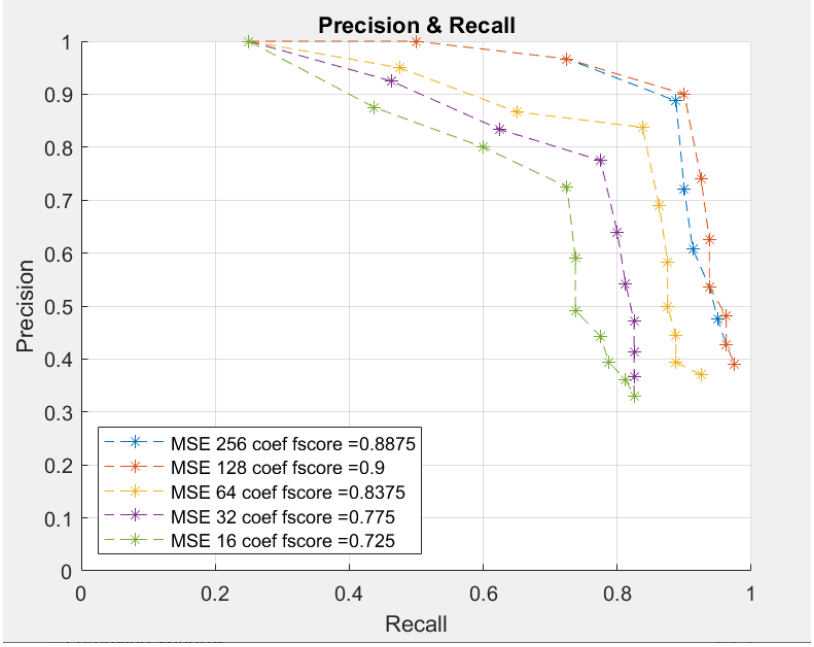


Gràfica 2: *Precision-Recall obtinguts fent servir histogrames de 256 bins i 256 coeficients (sense compressió) i quantificació logarítmica i lineal.*

Com podem veure a la gràfica els resultats de la compressió logarítmica a nivell de fscore són significativament millors que els de la compressió lineal, independentment de la distància escollida.

3. Efectes de reduir el nombre de bins HSV vs reduir el nombre de coeficients de Haar

Per poder comprovar l'efecte que té sobre la cerca de candidats el fet de de reduir el nombre de bins de l'histograma HSV o el nombre de coeficients de Haar, hem fet taules comparatives. Fixant primerament el número de bins al màxim 256 i variant el número de coeficients.



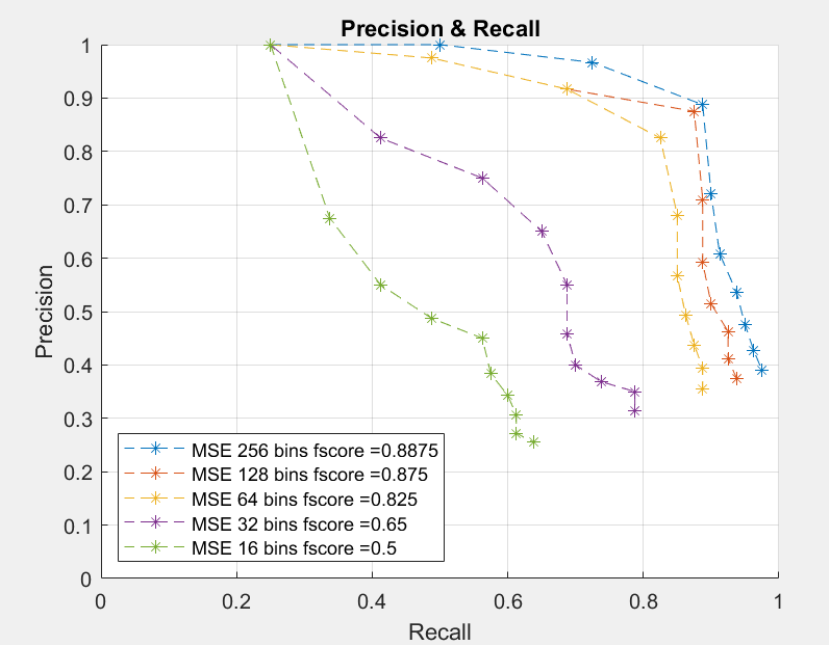
Gràfica 3: *Precision-Recall obtinguts fent servir histogrames de 256 bins i un nombre variable de coeficients, quantificació logarítmica i distància MSE.*

Taula 2: Fscore i temps/imatge de cada distància per cada nombre de coeficients, sempre amb 256 bins.

# Coeficients	Distància	Temps total (s)	Temps/imatge (s)	Fscore (%)
256	MAE	3,0128	0,15064	0,8875
	'RMAE'	3,6479	0,182395	0,85
	MSE	3,6463	0,182315	0,8875
128	MAE	1,9564	0,09782	0,8875
	'RMAE'	3,82	0,191	0,85
	MSE	2,1771	0,108855	0,9
64	MAE	1,3568	0,06784	0,85
	'RMAE'	2,7101	0,135505	0,775
	MSE	1,5533	0,077665	0,8375
32	MAE	1,4656	0,07328	0,775
	'RMAE'	1,3002	0,06501	0,7625
	MSE	1,1083	0,055415	0,775
16	MAE	0,8784	0,04392	0,675
	'RMAE'	0,9165	0,045825	0,6875
	MSE	1,0142	0,05071	0,725

Com podem veure a la taula, els valors més elevats de fscore s'aconsegueixen amb 256 i 128 coeficients, obtenint-se el millor fscore per la distància MSE amb 128 coeficients. La reducció del nombre de coeficients afecta als resultats de fscore, que arriba a reduir-se fins a 0'2 punts, però segueix donant resultats bastant bons i ajuda a disminuir en gran part el temps d'execució.

En segon lloc, fem els càlculs fixant els coeficients al màxim disponible (no iterar amb Haar) i variant el nombre de bins de punt de partida.



Gràfica 4: *Precision-Recall obtinguts fent servir histogrames de nombre de bins variable, el màxim de coeficients en cada cas, quantificació logarítmica i distància MSE.*

Taula 3: Fscore i temps/imatge de cada distància per cada nombre de bins, mantenint els coeficients.

# Bins	Distància	Temps total (s)	Temps/imatge (s)	Fscore (%)
256	MAE	3,0128	0,15064	0,8875
	'RMAE'	3,6479	0,182395	0,85
	MSE	3,6463	0,182315	0,8875
128	MAE	2,6826	0,13413	0,875
	'RMAE'	2,5301	0,126505	0,8375
	MSE	2,969	0,14845	0,875
64	MAE	1,2803	0,064015	0,8375
	'RMAE'	1,4782	0,07391	0,8
	MSE	1,7876	0,08938	0,825
32	MAE	1,068	0,0534	0,675
	'RMAE'	1,3403	0,067015	0,625
	MSE	1,0566	0,05283	0,65
16	MAE	0,8365	0,041825	0,488889
	'RMAE'	0,8564	0,04282	0,45
	MSE	0,9448	0,04724	0,5

Com podem veure a la taula, els valors més elevats de fscore s'aconsegueixen amb 256, 128 i fins i tot 64 bins, obtenint-se el millor fscore per la distància MSE i MAE amb 256 bins. La reducció del número de bins afecta encara més als resultats de fscore del que ho feia la reducció dels coeficients, ja que arriba a reduir-se fins a més de 0'4 punts.

Per acabar aquesta comparació, hem decidit comparar també la temporització obtinguda reduint coeficients vs la temporització obtinguda reduint bins i examinar l'ocupació dels fitxers generats.

Taula 4: temps total i temps/imatge segons el nombre de coeficients i nombre de bins.

# coeficients final	Temps total reduint coeficients (s)	Temps total reduint bins (s)	Temps/imatge reduint coeficients (s)	Temps/imatge reduint bins (s)
256	53,2982	52,0053	0,0266491	0,02600265
128	58,3361	45,8484	0,02916805	0,0229242
64	63,686	42,3193	0,031843	0,02115965
32	74,6255	43,8264	0,03731275	0,0219132
16	77,5273	41,0638	0,03876365	0,0205319

Com veiem a la taula, la reducció de coeficients provoca un increment tant del temps total com del temps per imatge. La reducció del nombre de bins, per contra, provoca una disminució de la temporització.

Taula 5: ocupació en memòria segons el nombre de coeficients i nombre de bins.

# coeficients final	Bytes totals	Bytes totals reduint bins	Bytes/imatge reduint coeficients	Bytes/imatge reduint bins
256	1267489	1267489	634	634
128	658885	656312	329	328
64	353653	337612	177	169
32	201822	161967	101	81
16	107373	84364	54	42

L'espai en memòria dels nostres fitxers de base de dades, coherentment amb el procediment de càlcul, es redueix en 2 cada vegada que reduïm el nombre de coeficients final. Es pot observar també que en comprimir bins obtenim una mida menor que en reduir coeficients, possiblement causat per la pèrdua d'informació (bins buits).

Anàlisi dels Resultats i Conclusions

L'estudi realitzat en l'extracció de resultats ens ha permès trobar la millor configuració per el nostre comparador prog2. A sota es mostra una taula amb les millors configuracions de prog1 i prog2, on es pot veure que, malgrat que el temps per imatge de prog2 és bastant més elevat que el de prog1, el fscore obtingut és gairebé el doble.

Taula 6: taula resum sistemes prog1 i prog2 amb la millor configuració

Sistema	Distància	# bins	# coeficients	Temps/imatge (s)	Bytes/imatge	Fscore
Histograma de nivells de gris	'RMAE'	16	-	0,033855	79	0,55
SCD	MSE	256	128	0,108855	329	0,9

1. Anàlisi del número de bins / número de coeficients

1.1. fscore

Reduir bins té un efecte molt pitjor sobre el fscore que reduir coeficients. Això es deu al fet que en agafar menys bins, estem fent una primera quantificació de la imatge pitjor, amb menys bits, i per tant perdent informació. En canvi, en disminuir el nombre de coeficients, el que estem fent és sumar entre ells els coeficients anteriors, de forma que encara que en aquesta suma perdem detall, no perdem tanta informació com ho fèiem en quantificar amb menys bins d'inici.

1.2. temporització

La reducció del nombre de bins, provoca una disminució de la temporització. Això és coherent amb el fet que si quantitzem la imatge inicialment amb un histograma de menys bins, després la resta de càlculs (quantitzacions diverses, càlcul de coeficients) es fan sobre menys valors, disminuint el temps necessari pel processat. La reducció de coeficients, per contra, provoca un increment de la temporització. Això és coherent amb el fet que reduir coeficients implica fer més sumes de bins durant el càlcul de la transformada de Haar.

1.3. Conclusió

Reduir bins permet accelerar la generació de la base de dades, però perjudica els resultats. Per tant en la nostra aplicació, en la que generar la base un cop és suficient, no compensa el guany en temps a la pèrdua de qualitat fscore.

Reduir coeficients és una bona manera de reduir l'ocupació en memòria sense degradar excessivament el fscore. La implementació de la transformada de Haar fa que l'efecte de reduir coeficients resulti en una compactació de la informació, amb una pèrdua petita o mínima: reduir fins 128 o 64 coeficients segueix donant molt bons resultats.

2. Anàlisi de les distàncies i comparació entre prog1 i prog2

Les tres distàncies es basen en trobar la diferència entre cadascun dels coeficients calculats: el MAE, en calcula la diferència en valor absolut, el MSE en calcula la diferència al quadrat i el 'RMAE' l'obté al fer l'arrel quadrada de la diferència. És a dir, el MSE emfatitza els errors grans entre coeficients i discrimina els petits respecte el MAE i el 'RMAE' suavitza els errors grans.

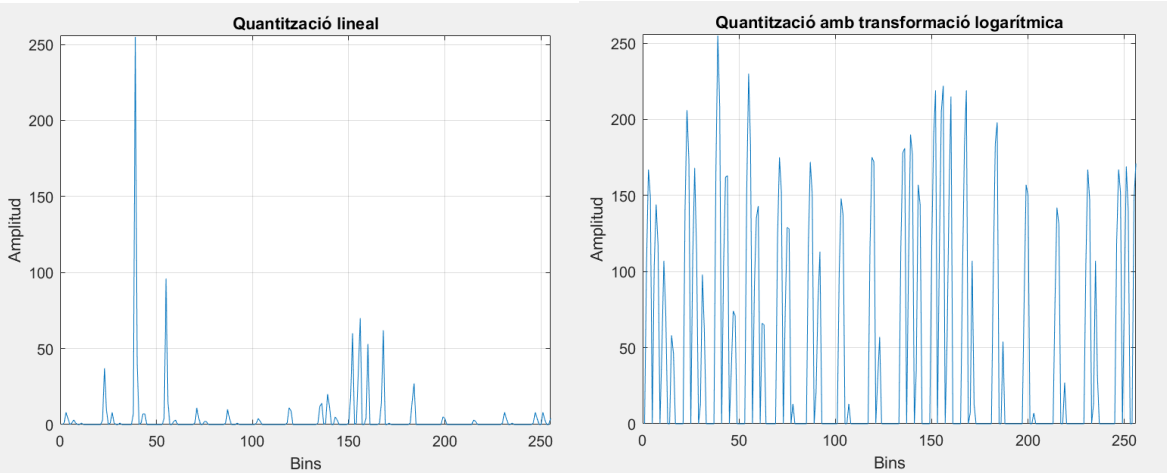
Al treballar amb aquests coeficients, tot i que les distàncies donen resultats molt similars, surt més a compte prioritzar errors grans (MSE) que discriminar-los ('RMAE'), situació contrària al que ens vam trobar en prog1 quan treballàvem sobre bins de nivell de gris directament. Part d'això es causa pel fet que ara els coeficients contenen molta més informació de la imatge (to, saturació i brillantor) que en prog1 (to de gris) i, per tant, errors grans al comparar coeficients haurien d'estar més penalitzats, ja que indiquen un canvi que pot contenir informació de més d'un dels paràmetres de la imatge.

A comentar que ens ha sobtat que generalment fer els càlculs utilitzant la distància MSE és més ràpid que no pas fer-ho amb MAE. Això ens porta a pensar que per MatLab és més costós calcular un valor absolut que no pas el quadrat d'una diferència.

*Les distàncies s'han escollit basant-nos en propostes de papers sobre distàncies i processat d'imatge [2][3][4].

3. Anàlisi de la compressió

La quantització amb transformació logarítmica dóna molt millors resultats que la quantització lineal. Això es deu a que la normalització de l'histograma es fa dividint els nivells entre el nombre total de píxels de la imatge, de forma que encara que el rang de valors possibles de sortida vagi de [0,1] és poc probable que tots els píxels o una gran quantitat vagin a parar a un mateix nivell. Per tant tindrem més valors propers a 0 que a 1, amb lo qual ens interessa una compressió que tingui més nivells per valors propers a 0 ---> compressió logarítmica. Podem veure aquesta explicació de forma més gràfica si mirem el resultat de la quantificació lineal i logarítmica de l'histograma normalitzat de la imatge ukbench00000.jpg.



Imatge 2: *Histogrames comparatius de quantificació lineal i logarítmica de la imatge ukbench 00000.jpg.*

4. Possibles Millores

Com a possible millora proposem una eina que permetés tenir en compte la posició relativa dels colors dominants (per exemple funcionar per blocs). Això arreglaria que imatges amb colors similars però situats de forma molt diferent els uns respecte als altres, no es confonguessin com a la mateixa imatge.

5. Dificultats

La principal dificultat que ens hem trobat a l'hora de realitzar aquest treball ha estat entendre l'espai de color HSV. El concepte de donar més pes (nombre de bits) a una variable (H, S, V) que a l'altra i entendre que s'havien de comprimir en una sola per generar l'histograma.

Bibliografia

[1] Jens-Rainer Ohm, Leszek Cieplinski, Heon J. Kim, Santhana Krishnamachari, B. S. Manjunath, Dean S. Messing and Akio Yamada. *Color Descriptors*.
[2] Alexei Botchkarev. Performance Metrics (Error Measures) in Machine Learning Regression, Forecasting and Prognostics: Properties and Typology
[3] Palubinskas, Gintautas. (2016). Image similarity/distance measures: what is really behind MSE and SSIM?. International Journal of Image and Data Fusion. 8. 1-22. 10.1080/19479832.2016.1273259.
[4]https://towardsdatascience.com/deep-learning-image-enhancement-insights-on-loss-function-engineering-f57ccbb585d7

Annex

Codi Matlab

algo1.m

```
function algo1(nbins)
%outputs histogram_database.txt file, which contains the monochrome
%histograms used to compare images
%  nbins: # of bins of the histograms

%% DECLARE VARIABLES
dir = pwd;      %set directory to the current one
cd(dir);
cd("database")  %go to the directory of the image database
N = 2000;       %# of images

%% READ IMAGES AND WRITE THEM INTO THE OUTPUT FILE
fileID = fopen('../histogram_database.txt','w');
for id = 0:N-1
    %obtain image name and retrieve its file
    num = sprintf('%05d', id);
    nameim = strcat('ukbench', num , '.jpg');

    img = imread(nameim);   %read image
    imgrey = rgb2gray(img); %obtain grayscale image
    h = round(imhist(imgrey, nbins));   %obtain histogram h

    %print bins into histogram_database.txt
    fprintf(fileID,'%d ',h);
    fprintf(fileID,'\n');
end
%close file and return to the main directory
fclose(fileID);
cd('..')
```

algoHSV.m

```
function algoHSV(coefficients, nbins, quantization)
%outputs histogram_database.txt file, which contains the approximation
%coefficients used to compare images
%   coefficients: # of haar coefficients kept
%   nbins: # of bins used for the hsv histogram
%   quantization: choose whether first the quantization is:
%       -linear (quantization=='linear')
%       -logarithmic (quantization=='log')

%% DECLARE VARIABLES
dir = pwd;      %set directory to the current one
cd(dir);
cd("database")  %go to the directory of the image database
N = 2000;       %# of images

%% READ IMAGES AND WRITE THEM INTO THE OUTPUT FILE
fileID = fopen('../histogram_database.txt','w');
for id = 0:N-1
    %obtain image name and retrieve its file
    num = sprintf('%05d', id);
    nameim = strcat('ukbench', num , '.jpg');
    img = imread(nameim);   %read image
    hsvImage = rgb2hsv(img); %convert values RGB->HSV

    %generate coefficients for every image
    [quantized_coefficients] = generateHSVhist(hsvImage, coefficients, nbins,
quantization);
```

```
    %print coefficients into histogram_database.txt
    fprintf(fileID,'%d ', quantized_coefficients);
    fprintf(fileID,'\n');

end
%close file and return to the main directory
fclose(fileID);
cd('..')
```

generateHSVhist.m

```
function [aprox_coef] = generateHSVhist(hsvImage, coefficients, nbins, quantization)
%generates the quantized haar coefficients of a hsv histogram of nbins,
%which has been previously quantized linearly (quantization=='linear') or
%nonlinearly (quantization=='log')
%   hsvImage: matrix containing hsv values of a given image
%   coefficients: # of haar coefficients kept
%   quantization: choose whether first the quantization is:
%       -linear (quantization=='linear')
%       -logarithmic (quantization=='log')

%% DECLARE VARIABLES
% select bit amount accordingly
switch nbins
    case 256
        h_levels = 16;                                  % # of hue levels
        s_levels = 4;                                   % # of saturation levels
        v_levels = 4;                                   % # of value levels
    case 128
        h_levels = 8;                                   % # of hue levels
        s_levels = 4;                                   % # of saturation levels
        v_levels = 4;                                   % # of value levels
    case 64
        h_levels = 8;                                   % # of hue levels
        s_levels = 2;                                   % # of saturation levels
        v_levels = 4;                                   % # of value levels
    case 32
        h_levels = 8;                                   % # of hue levels
        s_levels = 2;                                   % # of saturation levels
        v_levels = 2;                                   % # of value levels
    case 16
        h_levels = 4;                                   % # of hue levels
        s_levels = 2;                                   % # of saturation levels
        v_levels = 2;                                   % # of value levels
end
h_bits = log2(h_levels);                                  % # of hue bits
s_bits = log2(s_levels);                                  % # of saturation bits
v_bits = log2(v_levels);                                  % # of value bits
total_bits = h_bits+s_bits+v_bits;                      % total bits

hsvHist = zeros(nbins,1);                % HSV histogram array

bits_per_hist_level = 8; % #bits to quantize the # of pixels of a histogram level (0..255)

haarHist=zeros(1,coefficients); % haar coefficients array

%% GENERATE HISTOGRAM
[numRows, numCols, ~] = size(hsvImage); % retrieve height and width of the image
total_pixels = numRows*numCols;         % obtain total pixels

% The histogram (in the HSV color space) values are extracted %
for col = 1 : numCols
    for row = 1 : numRows
            %hsvImage has values that range from 0 to 1. We must adapt
            %their dynamic range to a value between 0 ... h/s/v_levels-1
```

```
            hBin = ceil(hsvImage(row, col, 1) * (h_levels-1));  % adapt hue range
            sBin = ceil(hsvImage(row, col, 2) * (s_levels-1));  % adapt saturation range
            vBin = ceil(hsvImage(row, col, 3) * (v_levels-1));  % adapt value range

            % each pixel value is a value that contains information about
            % the three histogram bins (h,s,v):
            % The h_bits most significant bits are the hue
            % The s_bits following bits are the saturation
            % The v_bits least significant bits are the value

           pixel = 2^(total_bits-h_bits)*hBin+2^(total_bits-h_bits-s_bits)*sBin+vBin;

            % Add 1 to the value of the pixel
            % since matlab indexes start at 1 and our values range from 0 to level-1,
            % a unit is added to the array index to make up for that
            hsvHist(pixel+1) = hsvHist(pixel+1)+1;

    end
end

%% LINEAR/NONLINEAR QUANTIZATION
% Normalize histogram values
hsvHist = hsvHist/total_pixels; %Returns values between 0 and 1

% According to the user's selection, the quantization will be linear or
% nonlinear
switch quantization
    case 'linear'
        % Linear quantization of coefficients
        qHist = round(hsvHist/max(hsvHist)*(pow2(bits_per_hist_level)-1));
    case 'log'
        % Logarithmic quantization
        hsvHist_log = 10*log10(hsvHist+10^-5);
        % obtain max value of histogram and quantize linearly
        % due to the logarithm and our normalised values, hsvHist_log
        % contains <0 values, which makes the linear quantization is a bit tricky
        % (the lowest value is sustracted, which will set it to 0 and
        % increase the 'less negative' ones accordingly ->(hsvHist_log-min(hsvHist_log))
        qHist = round((hsvHist_log-min(hsvHist_log))/max(hsvHist_log-
min(hsvHist_log))*(pow2(bits_per_hist_level)-1));
end

%% HAAR COEFFICIENTS RETRIEVAL
% if the coefficients are higher (an unreasonable decision) or equal to the # of histogram
bins,
% haar won't be used, since it would decrease the # of haar coefficients
if(coefficients>=nbins)
    haarHist = qHist;
else
    %last_j = num_histogram_bins-num_histogram_bins/coefficients-1
    %first_j = 1
    %j increment = num_histogram_bins/coefficients = iter
    index = 1;  % index to fill the resulting array
    last_j=nbins-nbins/coefficients-1;
    iter = nbins/coefficients; % amount of values that have to be
    %accumulated (eg: for 128 coefficients, two sums will be accumulated in each
    %bin -> 1+2, 3+4, ...127+128)

    %iterates through the entire quantized histogram with iter skips
    for j=1:iter:last_j
        %performs the local sums
        for count=1:iter
            haarHist(index)= haarHist(index)+qHist(count+j);    %store valeus into
haarHist(index)
        end
```

```
        index = index+1;    %gets increased every iter times
    end
end

%% LINEAR QUANTIZATION
% linear quantization of the coefficients
aprox_coef = round(haarHist/max(haarHist)*(pow2(bits_per_hist_level)-1));
end
```

algo2.m

```
function [obtained_dist] = algo2(id1,id2,data_matrix, distance)
% returns the distance between two images
%   id1: id of the first image
%   id2: id of the second image
%   data_matrix: matrix containing all the values of histogram_database.txt
%   distance: distance used for the measurement

%% OBTAIN ID1 AND ID2 COEFFICIENTS
hist1 = data_matrix(id1+1, :);
hist2 = data_matrix(id2+1, :);
dist_sum = 0;
N = length(hist1);

%% COMPUTE DISTANCE GIVEN THE USER INPUT
switch distance

    case 'rmae'
        for i=1:N
            dist_sum = dist_sum +sqrt(abs((hist1(i)-hist2(i))));  % "RMAE"
        end
    case 'mse'
        for i=1:N
            dist_sum = dist_sum +(hist1(i)-hist2(i)).^2;          % MSE
        end
    case 'mae'
        for i=1:N
            dist_sum = dist_sum +abs((hist1(i)-hist2(i)));        % MAE
        end
end

dist_sum = dist_sum/N;
obtained_dist = dist_sum;

end
```

algo3.m

```
function algo3(query,output_filename, input_filename, candidates, user_root, distance)
% given a query image, searches the most likely candidates and prints them
% into an output file
%   query: query image filename
%   output_filename: output txt filename
%   candidates: # of candidates that will be retrieved
%   user_root: current directory
%   distance: distance used for the measurement

%% EXTRACT QUERY ID
s = split(query, "ukbench0");
s = split(s(2), ".jpg");
query_id = str2num(s(1));   % convert to a scalar

%% OBTAIN ALL COEFFICIENTS AND OPEN THE NECESSARY FILES
input = dlmread(input_filename);

% get the number of images to be queried
```

```
num_images = length(input(:,1));

% open output file to write the results
a=fopen([user_root, '/', 'output_temp.txt'],'w');
b = fopen([user_root, '/', output_filename],'a');
if(a==-1||b==-1)
    printf("Couldn't open file \n");
    return
end

%% OBTAIN MOST SIMILAR IMAGES AND WRITE IT INTO THE OUTPUT FILE
distance_array = (1:num_images);
%compute distances of query with all images on database
for i=1:num_images
    distance_array(i) = algo2(query_id, i-1, input, distance);
end

% select the most similar images
[~, Similar_images] = mink(distance_array, candidates);

% write the results into the output file
fprintf(a,'Retrieved list for query image %s \n',query);
fprintf(b,'Retrieved list for query image %s \n',query);
for j=1:candidates
    fprintf(a,'%s\n',sprintf('ukbench%05d.jpg',Similar_images(j)-1));
    fprintf(b,'%s\n',sprintf('ukbench%05d.jpg',Similar_images(j)-1));
end

fprintf(b,'\n');
% close files
fclose(a);
fclose(b);
```

p_r.m

```
function [precision, recall] = p_r
% returns the precision and recall values for one given image
%   query_image: queried image for the measurement

%% OBTAIN RETRIEVED IDS
% obtain candidates of the temporary file
out = readmatrix('output_temp.txt', 'OutputType','string');

% extract image ids
for i=1:length(out)
    s = split(out(i), "ukbench0");
    s = split(s(2), ".jpg");
    out_images_id(i) = str2num(s(1));
end

%% COMPUTE PRECISION AND RECALL VALUES
relpos = mod(out_images_id(1),4); % informs of relative position of image
startpos = out_images_id(1) - relpos; % returns the first absolute image id of the 4 image
set

TP=0; % true positive count

candidates = length(out_images_id)-1;   % # of candidates

precision = zeros(1, candidates);       % precision empty array
recall = zeros(1, candidates);          % recall empty array

for i=2:candidates+1
    if (out_images_id(i)== startpos || out_images_id(i)== startpos + 1 ||
out_images_id(i)== startpos + 2 || out_images_id(i)== startpos + 3)
        % update TP count
```

```
        TP= TP+1;
    end
    % precision: TP/(TP+FP)
    precision(i-1) = TP/(i-1);
    % recall: TP/(TP+FN)
    recall(i-1) = TP/(4);   % total positives==4 in this experiment
end
end
```

main.m

```
function fmeasure = main(varargin)
%Computes the fmeasure and plots the precision & recall curve
%USAGE:
%   case 1 input argument:
%       fmeasure = main(distance_method)
%       -> computes fmeasure using distance_method given an input file (input.txt)
%
%       case 2 input arguments:
%       fmeasure = main(distance_method, arg2)
%       -> if arg2=='all':computes fmeasure using distance_method of all
%       images
%       -> if arg2=number: computes fmeasure of using distance_method of
%       /number/ randomly selected images
%
%       distance_method: measurement used to obtain the results.
%       Options are:
%           -MAE (distance_method=='mae')
%           -RMAE (distance_method=='rmae')
%           -MSE (distance_method=='mse')


%% DECLARE VARIABLES
% Directory/file-related variables
Output_filename= 'output.txt';
User_root      = pwd;
Input_filename = 'histogram_database.txt';  %input file

Candidates     = 10;  % Number of candidates to retrieve

avg_pre = zeros(1,Candidates);  % create empty precision array
avg_rec = zeros(1,Candidates);  % create empty recall array

% check for an output file
% (since we rely on writing in append mode, we must make sure the output
% file is empty by deleting it)
if(isempty(dir(Output_filename))==0)
    delete (Output_filename);
end


if nargin==1
%% CASE: IDS RETRIEVED FROM AN INPUT FILE
    num_queries=20;         % set query #
    distance=varargin{1};   % assign first and only argument to distance

    %read input file and store it in in_array
    in_array = readmatrix('input.txt', 'OutputType','string');

    %for every query
    for i=1:num_queries
        % obtain query name
        query_name=in_array(i);

        % generate candidates using the selected distance and write them into
```

```
        % the output file
        algo3(query_name,Output_filename, Input_filename, Candidates, User_root, distance);

        % compute precision and recall values
        [precision, recall] = p_r;
        avg_pre = avg_pre+precision;
        avg_rec = avg_rec+recall;
    end
elseif nargin==2
%% CASE: RANDOMLY GENERATED IDS OR ALL IDS
    distance=varargin{1};   % assign first argument to distance

    %check the second argument and set num_queries accordingly
    if(varargin{2}=='all')
        num_queries =2000;
    else
        num_queries=varargin{2};
    end

    for i=1:num_queries
        % if 'all' is selected, all ids will be swept (0...1999)
        if(varargin{2}=='all')
            query_id=i-1;
        % if a number is inputted, that number of ids will be randomly
        % generated
        else
            query_id = random('Discrete Uniform',1999);
        end

        % generate file name with the given id
        s =  sprintf('%05d', query_id);
        query_name = strjoin([ "ukbench", s ,".jpg"],'');

        % generate candidates using the selected distance and write them into
        % the output file
        algo3(query_name,Output_filename, Input_filename, Candidates, User_root, distance);

        % compute precision and recall values
        [precision, recall] = p_r;
        avg_pre = avg_pre+precision;
        avg_rec = avg_rec+recall;

    end
end

%% COMPUTE F-MEASURE AND CURVES
% obtain average precision and recall
avg_pre = avg_pre./num_queries;
avg_rec = avg_rec./num_queries;

% compute fmeasure and keep the maximum value
fmeasure= 2 * avg_pre.*avg_rec./(avg_pre+avg_rec);
fmeasure=max(fmeasure);

% delete temporary file
delete 'output_temp.txt'

%% PLOT CURVES
title('Precision & Recall')
xlabel('Recall')
ylabel('Precision')
grid on
hold on
plot(avg_rec, avg_pre, '--*')
```

```
xlim([0 1])
ylim([0 1])

end
```

Resultats generats a partir de `input.txt`

```
Retrieved list for query image ukbench01701.jpg
ukbench01701.jpg
ukbench01703.jpg
ukbench01700.jpg
ukbench01702.jpg
ukbench01808.jpg
ukbench01809.jpg
ukbench01740.jpg
ukbench01527.jpg
ukbench01743.jpg
ukbench01826.jpg

Retrieved list for query image ukbench00926.jpg
ukbench00926.jpg
ukbench00924.jpg
ukbench01410.jpg
ukbench01408.jpg
ukbench00919.jpg
ukbench00927.jpg
ukbench00917.jpg
ukbench00918.jpg
ukbench01409.jpg
ukbench00925.jpg

Retrieved list for query image ukbench01883.jpg
ukbench01883.jpg
ukbench01880.jpg
ukbench01881.jpg
ukbench01882.jpg
ukbench00098.jpg
ukbench00096.jpg
ukbench00204.jpg
ukbench00208.jpg
ukbench00080.jpg
ukbench00630.jpg

Retrieved list for query image ukbench00116.jpg
ukbench00116.jpg
ukbench00117.jpg
ukbench00118.jpg
ukbench00119.jpg
ukbench00080.jpg
ukbench00098.jpg
ukbench00947.jpg
ukbench00956.jpg
ukbench00096.jpg
ukbench01880.jpg

Retrieved list for query image ukbench00213.jpg
ukbench00213.jpg
ukbench00215.jpg
ukbench00214.jpg
```

```
ukbench01083.jpg
ukbench01063.jpg
ukbench01235.jpg
ukbench01233.jpg
ukbench00663.jpg
ukbench01062.jpg
ukbench01080.jpg

Retrieved list for query image ukbench00693.jpg
ukbench00693.jpg
ukbench00695.jpg
ukbench00692.jpg
ukbench00694.jpg
ukbench00315.jpg
ukbench00313.jpg
ukbench00312.jpg
ukbench00189.jpg
ukbench00191.jpg
ukbench00188.jpg

Retrieved list for query image ukbench00379.jpg
ukbench00379.jpg
ukbench00378.jpg
ukbench00376.jpg
ukbench00377.jpg
ukbench00380.jpg
ukbench00165.jpg
ukbench00399.jpg
ukbench00849.jpg
ukbench00383.jpg
ukbench00167.jpg

Retrieved list for query image ukbench00466.jpg
ukbench00466.jpg
ukbench00464.jpg
ukbench00465.jpg
ukbench00467.jpg
ukbench00138.jpg
ukbench00137.jpg
ukbench00553.jpg
ukbench01681.jpg
ukbench00922.jpg
ukbench00554.jpg

Retrieved list for query image ukbench00600.jpg
ukbench00600.jpg
ukbench00601.jpg
ukbench00609.jpg
ukbench00603.jpg
ukbench00602.jpg
ukbench00608.jpg
ukbench00610.jpg
ukbench00685.jpg
ukbench00568.jpg
ukbench00611.jpg

Retrieved list for query image ukbench00458.jpg
ukbench00458.jpg
ukbench00456.jpg
```

```
ukbench00459.jpg
ukbench00457.jpg
ukbench00678.jpg
ukbench00578.jpg
ukbench00579.jpg
ukbench00679.jpg
ukbench01929.jpg
ukbench00676.jpg

Retrieved list for query image ukbench00771.jpg
ukbench00771.jpg
ukbench00770.jpg
ukbench00768.jpg
ukbench00769.jpg
ukbench00146.jpg
ukbench00147.jpg
ukbench00143.jpg
ukbench00145.jpg
ukbench00144.jpg
ukbench00142.jpg

Retrieved list for query image ukbench01776.jpg
ukbench01776.jpg
ukbench01777.jpg
ukbench01779.jpg
ukbench00513.jpg
ukbench00744.jpg
ukbench01900.jpg
ukbench00512.jpg
ukbench01778.jpg
ukbench01323.jpg
ukbench01489.jpg

Retrieved list for query image ukbench01507.jpg
ukbench01507.jpg
ukbench01505.jpg
ukbench01504.jpg
ukbench01691.jpg
ukbench01689.jpg
ukbench01496.jpg
ukbench01656.jpg
ukbench01688.jpg
ukbench01657.jpg
ukbench01651.jpg

Retrieved list for query image ukbench00440.jpg
ukbench00440.jpg
ukbench00443.jpg
ukbench00441.jpg
ukbench00442.jpg
ukbench01453.jpg
ukbench01321.jpg
ukbench01322.jpg
ukbench00517.jpg
ukbench00370.jpg
ukbench00369.jpg

Retrieved list for query image ukbench00903.jpg
ukbench00903.jpg
```

```
ukbench00901.jpg
ukbench00902.jpg
ukbench00900.jpg
ukbench00856.jpg
ukbench00857.jpg
ukbench00858.jpg
ukbench00792.jpg
ukbench00859.jpg
ukbench00815.jpg

Retrieved list for query image ukbench01350.jpg
ukbench01350.jpg
ukbench01351.jpg
ukbench01349.jpg
ukbench01448.jpg
ukbench01348.jpg
ukbench01450.jpg
ukbench01451.jpg
ukbench01449.jpg
ukbench01115.jpg
ukbench01248.jpg

Retrieved list for query image ukbench00804.jpg
ukbench00804.jpg
ukbench00806.jpg
ukbench00807.jpg
ukbench00843.jpg
ukbench00841.jpg
ukbench00842.jpg
ukbench00840.jpg
ukbench00805.jpg
ukbench00859.jpg
ukbench00835.jpg

Retrieved list for query image ukbench00601.jpg
ukbench00601.jpg
ukbench00600.jpg
ukbench00602.jpg
ukbench00603.jpg
ukbench00609.jpg
ukbench00610.jpg
ukbench00608.jpg
ukbench00716.jpg
ukbench00568.jpg
ukbench00615.jpg

Retrieved list for query image ukbench01998.jpg
ukbench01998.jpg
ukbench01997.jpg
ukbench01999.jpg
ukbench01996.jpg
ukbench01953.jpg
ukbench01995.jpg
ukbench01952.jpg
ukbench01994.jpg
ukbench01971.jpg
ukbench01993.jpg

Retrieved list for query image ukbench00801.jpg
```

ukbench00801.jpg
ukbench00802.jpg
ukbench00803.jpg
ukbench00800.jpg
ukbench00709.jpg
ukbench00708.jpg
ukbench01968.jpg
ukbench00253.jpg
ukbench00710.jpg
ukbench00784.jpg