ORGANITZACIÓ D'IMATGES SEGONS CONTINGUT

Memòria del Projecte Fi de Carrera
d'Enginyeria en Informàtica
realitzat per
Marc Vilana Roca
i dirigit per
Ricardo Toledo Morales
Bellaterra,.....de Juny de 2007

Universitat Autònoma de Barcelona

etse Escola Tècnica Superior d'Enginyeria

El sotasignat, **Ricardo Toledo Morales**

Professor/a de l'Escola Tècnica Superior d'Enginyeria de la UAB,

CERTIFICA:

Que el treball a què correspon aquesta memòria ha estat realitzat sota la seva direcció per en **Marc Vilana Roca**

I per tal que consti firma la present.

Signat: .....

Bellaterra, ...de Juny de 2007

INDEX

1.- Introducció

La imatge digital s'ha convertit en una tecnologia disponible en gairebé totes les llars, el seu fàcil ús i la versatilitat que ofereixen conjuntament amb la gran capacitat d'emmagatzematge d'avui dia han permès disposar d'enormes quantitats de material multimèdia, no sols en l'àmbit domèstic sinó en entorns especialitzats. El problema radica en com indexar aquestes enormes bases de recursos de forma eficient, els sistemes actuals d'organització no poden fer front quan parlem en aquest ordre, a part de la complexitat originada pel gran nombre de recursos multimèdia, sorgeix una nova necessitat, no sols volem poder cercar per nom o paraula clau, sinó que aprofitant els avantatges de la digitalització ens interessaria poder realitzar cerques mitjançant informació parcial de la imatge, com per exemple un objecte o utilitzar una imatge estàtica per a poder saber en quina sèrie, pel·lícula... apareix.

Cal doncs, un nou sistema que ens permeti més que organitzar, cercar de forma eficient contingut multimèdia, utilitzant com criteri de cerca paraules claus referents al contingut d'aquest. Tot i que aquesta solució implicaria realitzar una anotació manual de paraules claus "metadata" per a cada recurs, el que ho converteix en una solució prehistòrica i seria en tot cas una descripció subjectiva.

És en aquest entorn on neixen els sistemes CBIR. Aquests sistemes encara avui segueixen essent font d'investigació, no existeix un estàndard ni un conjunt de passos definits que donin una solució òptima al problema.

Aquesta memòria farà un repàs sobre les característiques més importants de les imatges (color, textura i forma). La combinació d'aquestes característiques seran les que ens donaran un índex de semblança entre imatges, s'utilitzaran diferents mètodes sobre una base de dades relativament petita, unes 500 imatges obtingudes aleatòriament i on com a mínim existeixen dues imatges semblants, per tal d'extreure algunes conclusions. Finalment s'implementarà una petita aplicació en Matlab que utilitzi els algorismes que millor resultat donen per aquesta base de dades.

En aquest tipus de problema la dificultat resideix en com mesurar el resultat, en aquest cas s'ha optat per utilitzar la detecció correcta així com la detecció de falsos

positius (cost molt petit) i falsos negatius (cost molt alt) com a paràmetres per mesurar el funcionament del programa. Utilitzant aquest criteri cal evitar caure en el parany de que el nombre de falsos positius sigui exageradament elevat, en aquest cas un usuari destinaria molt temps en discriminar els resultats. S'optarà també per incloure resultats d'aplicacions comercials sobre la nostra base de dades i comparar-ne el resultat.

Com a punt final recordar que al ser un problema obert no existeix encara una solució general òptima, per aquest motiu, els resultats obtinguts en aquest cas concret no necessàriament s'han de repetir amb una base de dades més gran o diferent.

1.1.- Objectius

Més enllà de la vessant pràctica que seria l'obtenció d'un prototip d'aplicació que doni resposta al problema inicial, es pretén veure si aquest tipus de solucions són aptes per a ser aplicades en la vida real i potser més important, a quin preu.

Ser un punt de partida per a futurs estudiants o persones interessades que comencen a endinsar-se en aquest món, de mode que puguin comprendre els principis d'aquests sistemes, i perquè no, millorar la solució proposada..

Entendre perquè, tot i ser un problema àmpliament treballat i investigat, encara no existeix una solució realment eficient i madura, que doni resposta, si més no, als escenaris d'aplicació més generals.

Obtenir les nostres pròpies conclusions sobre les diferents tècniques per afrontar aquests tipus de problemes. Realitzant tests sobre els algorismes més usuals i aquells que es consideren que tindran molta influencia en els pròxims anys.

Des d'un punt de vista de formació personal, s'espera assolir la metodologia necessària per a realitzar un projecte d'investigació. Familiaritzar-se amb les tècniques i els passos necessaris per fer front a una extensa allau d'informació, com és aquest cas. Sense oblidar les aportacions en quan a coneixements d'un tema que ja comença a ser bastant necessari, i sense cap mena de dubte, marcarà un punt i a part en la forma que cerquem la informació durant els pròxims anys.

1.2.- Definició

CBIR, sigles que fan referència a "Content Based Image Retrieval" (Extracció d'imatges segons contingut) es basa en obtenir les imatges segons les característiques principals d'aquestes, que son, com introduíem en el punt anterior, el color la textura i la forma. La motivació per implementar aquests mètodes s'ha mencionat també en la introducció, els mètodes actuals resulten obsolets. Fins ara s'utilitzava mètodes d'indexació basats en una paraula clau o bé un nombre. CBIR pretén anar més enllà i guardar per a cada imatge les seves característiques, d'aquesta manera una cerca consistirà en extreure les característiques de la imatge que li arriba com a consulta i comparar-la amb les característiques de les imatges de la base de dades, el mètode retornaria aquelles que tenen unes característiques més semblants. Com a punt final del mètode seria l'indexació mitjançant paraules, la consulta es realitzaria mitjançant una paraula, com per exemple, *poma*, cada imatge tindria associada un conjunt de paraules clau, es buscaria la imatge que respon a aquesta paraula clau i seria la que es passaria com a consulta, però com es deia és una solució rudimentària. Com realitzem la cerca doncs? Primerament fer una pinzellada en els tipus de consulta més generals, que són els següents:

- ***Consultes icòniques***: Aquest tipus de cerca s'utilitzarà quan es desitgi una cerca semàntica o d'alt nivell. Per aquest tipus de consultes és disposa d'icones que representen objectes o icones així com connectors lògics. L'usuari selecciona o construeix l'expressió de cerca mitjançant aquests connectors i seguidament un traductor l'analitza i la transforma.
- ***Consultes per dibuix:*** Destinat a usuaris amb coneixements de manipulació de color, en aquest cas l'usuari realitza un dibuix indicant les regions i el color que contindrà cada regió.
- ***Consultes per contorn:*** En aquest cas l'usuari utilitza com a petició un esborrany de l'objecte o situació que desitja obtenir.
- ***Consulta per imatge:*** Seria el cas més senzill, l'usuari introdueix una imatge del que desitja buscar. Aquesta cerca seria la més general, ja que comprèn totes les anteriors.

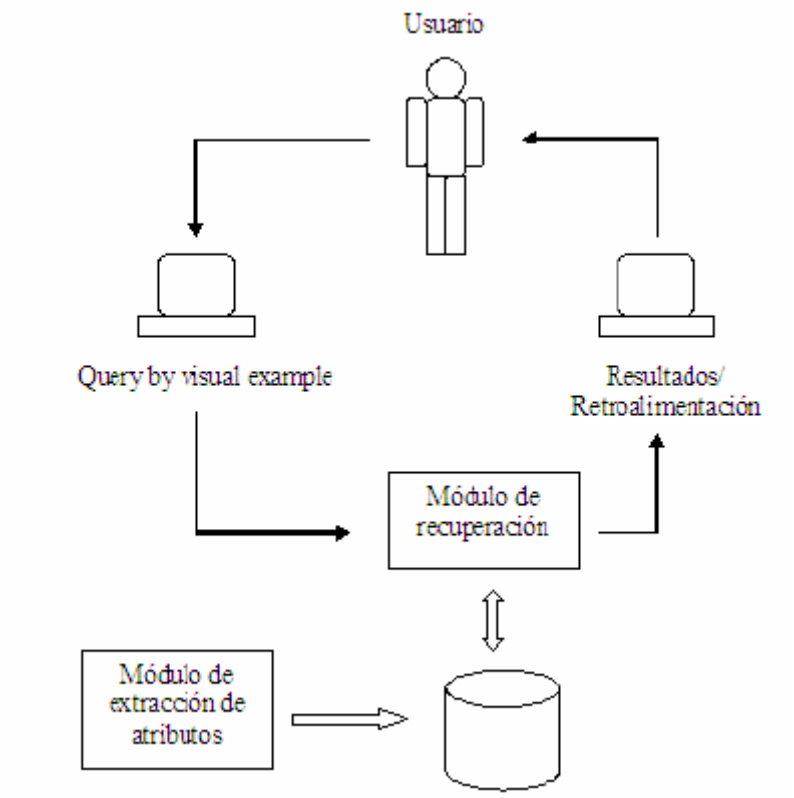
Per sistemes CBIR, l'última consulta resulta la més apropiada [5], ja que no depèn de l'habilitat que pugui tenir l'usuari, i no esta condiciona a un domini d'imatges de característiques molt especifiques. Per altra banda ens permet realitzar qualsevol tipus de cerca subjectiva, un cas exemple seria buscar un lloc relaxant, un usuari podria considerar que la platja és un paisatge relaxant i l'aplicació retornaria imatges de platja, per contra un altre usuari pot considerar que no ho és i en aquest cas utilitzaria una imatge d'una vall amb vegetació.

De forma abstracta es pot veure aquest mètode com un algorisme de dos passos, un primer pas que es coneix com a extracció de característiques i el que realitza es bàsicament el que el seu nom indica, buscar les característiques de cada imatge i guardar-les utilitzant alguna estructura de dades, habitualment acostumen a ser vectors. El segon pas, conegut com a "Matching" realitzaria la cerca en si, pretén comparar les característiques extretes de la imatge de petició amb les característiques de les imatges de la base de dades i retornar les més semblants. Aquesta comparació és realitza mitjançant el càlcul de la distància que separa les característiques.

Un sistema CBIR complet seguiria la figura1, on l'usuari a la vegada ajuda a millorar el sistema utilitzant un filtrat de les imatges que no l'interessaven, bàsicament consisteix en la discriminació dels falsos positius.

La figura 1 fa referència a l'esquema que hauria de seguir un sistema CBIR complert. Una primera etapa de processament de les dades que extreies una vegada les característiques de les imatges i les guardes a disc. Un usuari realitzaria la petició mitjançant una imatge de consulta, el mòdul de recuperació calcularia les característiques d'aquesta i les compararia amb les emmagatzemades a disc mostrant per pantalla els resultats. L'usuari llavors realitzaria una discriminació de les que no l'interessin adaptant i millorant d'aquesta forma l'aplicació.

Figura 1: ***Sistema CBIR complert.***



1.3.- Aplicacions

El camp d'aplicació d'aquest sistemes és gairebé infinit, i de forma general es pot dir que serà aquell on es necessiti indexar un gran nombre de imatges. Però en podríem destacar els següents:

- ***Prevenció criminal***: En aquest àmbit podria ser interessant disposar d'un reconeixedor de cares automàtic, per exemple podria estar en un aeroport i detectar si algun criminal esta intentant abandonar el país. Cal dir que els sistemes de reconeixement facial són molt madurs i obtenen resultats excel·lents.
- ***Verificació d'autenticitat:*** Per exemple en sistemes basats en detecció de l'empremta digital o escàners de retina.
- ***Diagnòstics mèdics:*** Detecció de possibles malalties a partir de les imatges obtingudes pels aparells mèdics, també permetria classificar segons la gravetat de la malaltia, aquest últim punt podria ser interessant per exemple en casos de llista de trasplantaments d'òrgans, organitzant de forma objectiva l'ordre de prioritat del trasplantament.

- ***Logotips de marques comercials:*** Podria ser útil quan per exemple una nova marca comercial vol registrar el seu logotip, entre altres, permetria detectar si aquest logotip esta registrat o no.
- ***Cerca en biblioteques digitals:*** Buscar la informació de forma ràpida i objectiva ja sigui en biblioteques locals o xarxes de biblioteques.

1.4.- Exemples de sistemes CBIR

Al ser un problema que pot ser aplicat en multitud de camps i en constant investigació, existeixen bastants exemples d'implementacions, en la majoria de casos, però són destinats a entorns concrets:

- ***GIFT:*** És un sistema CBIR de codi lliure, és el resultat de la investigació del grup de visió de la universitat de Gènova (Suïssa).
- ***IMGSEEK:*** Consisteix en un sistema CBIR distribuït sota llicencia GPL, com a característica destacar que el propi programa organitza automàticament les imatges del disc segons contingut.
- ***QBIC*** sistema CBIR desenvolupat per IBM, ***Virage***, ***IMatch*** desenvolupat per MWLabs i **ImageFinder** de Attrasoft. Aquestes aplicacions han estat dissenyades per a poder treballar amb bases d'imatges creades pels usuaris i per poder interactuar amb gestors de Bases de dades.
- ***VIR image engine:*** sistema CBIR semblant al QBIV desenvolupat per Virage.
- ***VisualSEEK*** i ***WebSEEK:*** Sistemes CBIR desenvolupats pel departament d'enginyeria elèctrica de la universitat de Colúmbia. Com a punt important destacar que VisualSeek compta amb l'extracció automàtica dels atributs de les imatges, utilitzant les regions de color més significatives. WebSEEK utilitza agents autònoms per a la cerca i recuperació de contingut multimèdia a la xarxa.
- ***NeTra:*** semblant a VisualSEEK però desenvolupat en aquest cas per la universitat de Califòrnia. Incorpora la possibilitat de buscar utilitzant les tres característiques de la imatge, de forma separada o amb combinacions d'elles. Com a complement a aquest sistema també han desenvolupat el sistema ***CIS*** que permet realitzar cerca textual.
- ***MARS:*** Desenvolupat per l'institut Beckman de ciència i tecnologia de la universitat de Il·linois.

- ***Blobworld:*** Aplicació desenvolupada per la universitat de Berkeley, en aquest cas esta enfocat en segmentar amb regions similars tant en color com en textura, el que li permet, de forma aproximada, distingir els objectes de les imatges.
- ***Viper:*** desenvolupar per la Universitat de Gènova, aquest exemple és interessant ja que es basa en la retroalimentació utilitzant un protocol de comunicació anomenat MRML (Multimèdia Retrieval Markup Language)
- A dia d'avui existeixen també eines de desenvolupament SDK que permeten crear aplicacions CBIR, l'aplicació ***Excalibur Visual Retrievalware Software*** en podria ser un exemple.

És una petita llista, però com es deia amb anterioritat existeixen molts sistemes que pretenen ser una possible solució al problema.

2.- Anàlisis del problema

Aquests tipus de problemes presenta una característica important, i és que tenen un cost computacional enorme. Ara bé, els resultats obtinguts per aquests mètodes son de lluny molt més acurats que els obtinguts per qualsevol altre mètode d'indexació actual. És fàcil adonar-se que existeix una relació entre el cost computacional i l'eficiència del mètode, mètodes més precisos requereixen un cost computacional més gran, ara bé a mesura que avança la potencia dels computadors ja sigui utilitzant multicomputadors o computadors domèstics aquesta relació es va fent cada cop menys important.

Qualsevol aplicació basada en aquest sistema segueix un algorisme genèric idèntic, utilitzar una imatge de query per extreure les característiques d'aquesta i comparar-les amb totes les de la base d'imatges. Les característiques són tres, color, textura i forma, que seran les que veurem seguidament.[8]

2.1.- Color

Una de les característiques més importants de tota imatge que fan possible el reconeixement pels humans és precisament aquesta, el color. Aquesta propietat depèn de la forma en que la llum és reflexa sobre l'objecte. Generalment el color es defineix en un espai tridimensional, conegut com a espai de representació del color. Existeixen tres representacions possibles, l'espai RGB ("Red" "Green" "Blue"), l'espai HSV ("Hue", "Saturation" "Value") i finalment el HSB ("Hue" "Saturation" "Brightness"), figura 2.

El primer espai seria una representació invariant de l'espai on cada color té el mateix pes, aquest espai és el que s'acostuma a utilitzar en imatge digital, els formats d'emmagatzematge JPEG,BMP,GIF, per exemple, l'utilitzen. Aquesta representació descriu els colors en base a tres components, cada un d'aquests components pot variar entre 0 i 255 el que ens proporciona un total de 16 milions de colors diferents.

Ara bé, el sistema humà no és perfecte, i per exemple distingim una quantitat major de colors vermellosos que blavosos, per aquest motiu apareixen els espais HSV i HSB. L'espai HSB també conegut com a espai Munsell, descriu els colors en base a tres

atributs, el matis que ens permet distingir el color; el valor que especifica la puresa d'un color, quantificat de 0 a 10 on el 0 equival al negre pur i el 10 al blanc pur. I la croma que ens indica la distancia en que es troba respecte al color neutre. L'espai HSV parteix del HSB però canvia la croma per la saturació, en aquest cas la saturació fa referència a la puresa del color, i oscil·la entre 0 i 100 % on els límits representen des d'ombres grises fins al color pur respectivament. En aquest espai el camp valor pren un altre significat, en aquest cas s'interpreta com la claredat del color.

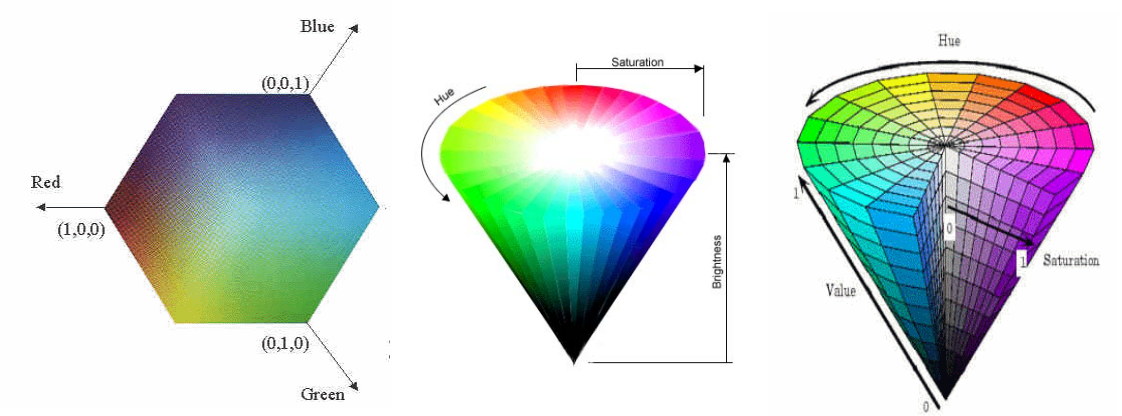


Figura 2: ***Comparativa dels tres espais, espai RGB, espai HSB i espai HSV***

2.1.1.- Histogrames de Color

La representació de color en sistemes CBIR s'acostuma a realitzar mitjançant histogrames de color, on sense entrar en detalls, dir que cada barra representa un color de l'espai de colors utilitzat i la seva freqüència representa el nombre de pixels associats a aquest color. S'aprecia un exemple en la figura 3.

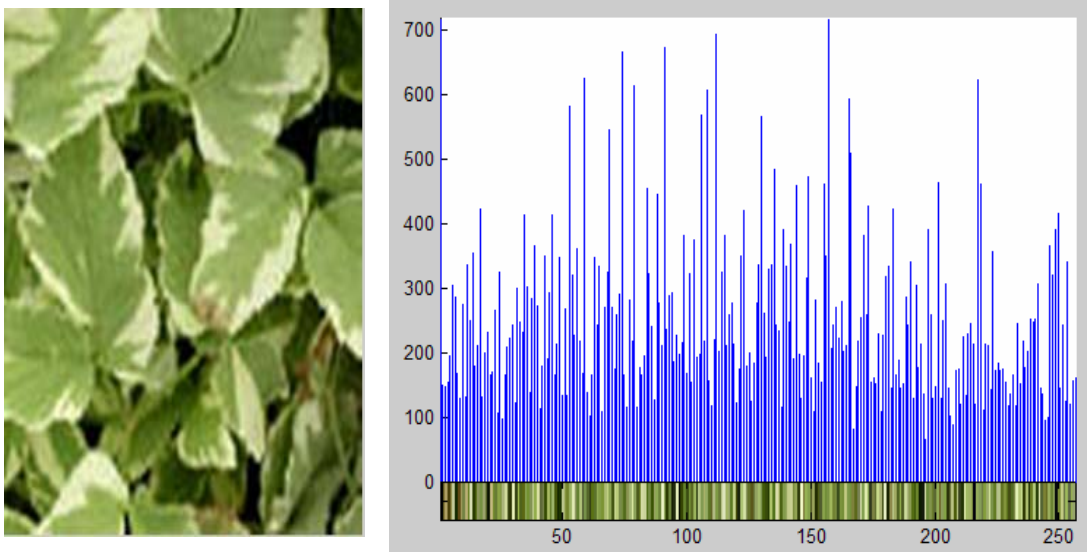


Figura 3: ***Una imatge amb el seu corresponent histograma de color***

Cal dir que existeixen dos tipus d'histogrames, els GCHs (Histogrames Globals) i els histogrames LCHs (Histogrames Locals).[12] Els histogrames globals representen la imatge amb un únic histograma de color, els histogrames locals per contra, divideixen la imatge en blocs i presenten l'histograma de color de cada un d'aquests blocs. En general els histogrames locals contenen més informació sobre la imatge que els histogrames locals, si bé aquests segons requereixen d'un temps de còmput més gran. És precisament aquest compromís entre representació de la imatge i temps de còmput el que converteix als histogrames de color globals com un dels mètodes clàssics en CBIR.

Un exemple d'histograma local seria la tècnica coneguda com a Grid9, consisteix en dividir una imatge en 9 regions de tamany uniforme. Per cada regió és calcula el seu histograma obtenint així una representació del color i una ubicació dins la imatge. Per a calcular la distància s'utilitza la mitjana de la distància quadràtica (3.2.1.1) de cada regió.

2.1.2.- Segmentació de la imatge

Com s'ha vist els histogrames globals ens fan perdre informació sobre la imatge, en alguns casos s'utilitza el que es coneix com a segmentació de la imatge, aquest mètode es basa bàsicament en dividir la imatge en regions o seccions, pel nostre cas concret utilitzarem una versió que es coneix com segmentació basada en regions, on el que pretén es buscar regions homogènies dins la imatge. Aquest mètode ens donarà bàsicament dos tipus d'informació, el nombre de colors dominants i l'àrea que aquests ocupen. Un cop obtinguda aquesta informació la comparació de color serà relativament trivial, les imatges amb el mateix nombre de colors dominants i àrees d'aquests properes en quan a quantitat seran imatges semblants. Un exemple d'execució es pot veure en la figura 4.

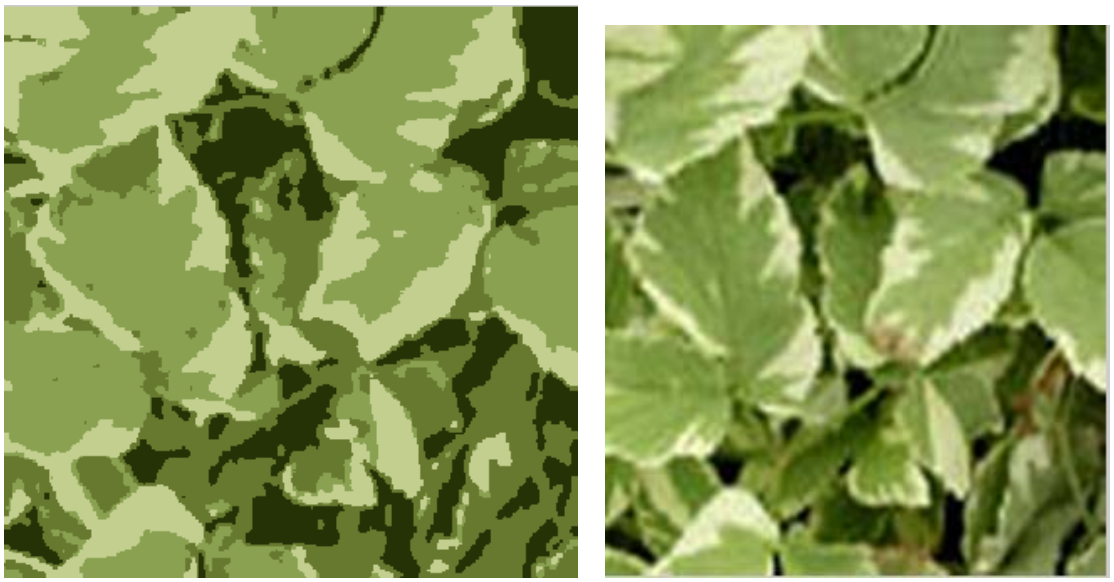


Figura 4: ***Imatge Segmentada i la seva imatge original.***

2.1.3.- Firmes Binàries

Aquest sistema fou proposat l'any 2002 per Nascimento i Chitkara. [4] Consisteix en representar les imatges mitjançant informació binària, cadenes de 0s i 1s. Aquestes cadenes pretenen representar la distribució dels colors existents dins la imatge. Fins ara, els mètodes vistos, plantejaven que tot color era igual d'important en una imatge, les firmes binàries trenquen amb aquesta idea i pretenen separar els colors dominants d'aquells que no ho són. Per diferenciar-los utilitza les característiques del sistema visual humà que segueix una corba sigmoïdal. L'algorisme segueix els següents passos:

1- Primerament hem de buscar els **n colors diferents**, de forma similar a com es realitza en els vectors de coherència (3.2.1.2.).
2- Un cop tenim els diferents colors els **dividim en vectors binaris** de longitud constant (CBA) o longitud variable (VBA). Cada vector acceptarà un cert rang de densitat d'aquest color. Després d'aquest pas obtindrem la firma binària que tindrà una expressió similar a la figura 5, on la n representa el color i la t el vector d'aquest color.

$$S = b_1^1 b_2^1 \ldots b_t^1 \;\; b_1^2 b_2^2 \ldots b_t^2 \ldots b_1^n b_2^n \ldots b_t^n$$

Figura 5: ***Expressió de la firma binària***

3- **Es calcula la densitat de cada color** dividint el nombre de pixels d'aquest color entre el nombre total de pixels de la imatge.
4- **S'assigna el valor 1** al bit b^i_j si la densitat del color i esta dins del rang que accepta el bit j, i **0 en cas contrari**.

El punt 3 i el punt 4 que resulten força confusos es veuran més clars en l'exemple proporcionat pels autors. Suposem la següent regió de la imatge de tamany 4x4, figura 6.

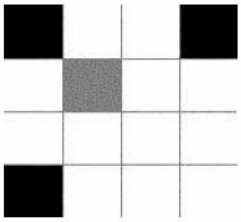


Figura 6: ***Imatge Exemple***

Aquesta regió conté tres colors. Suposem que fixem la distancia dels vectors amb valor t = 10. Calculem el vector de densitats de cada color. H = (3/16,1/16,12/16) i ens queda aproximadament H = (0.18, 0.06,0.75). Al fixar el valor t=10, estem fixant la tolerància a 100 /10 = 10 %. El que estem dient és que el primer vector acceptarà els valors entre (0 i 0.1], el segon entre (0.1 i 0.2] i així successivament amb els 10 vectors. Un cop tenim els vectors calculats hem de mirar en quin vector cau cada distribució de color. El color negre amb distribució 0.18 cau dins el segon vector, el color blanc caurà en el 8è vector, i el gris caurà dins el primer vector. Codifiquem aquesta assignació amb un 1 a la posició corresponent al nombre de vector. Ens queda que la codificació del negre és 0100000000, la del blanc és 0000000100, i la del gris 1000000000. La firma en aquest cas quedarà com S_a = 0100000000 0000000100 1000000000.

Com a detall mencionar que no té en compte la posició espacial dels colors. Finalment, un cop tenim calculada la firma per a dues imatges calculem la distància a la que es troben utilitzant la fórmula de la figura 7.

$$d(Q,I) = \sum_{j=1}^{n} \left[pos(B_Q^j) - pos(B_I^j) \right]^2$$

Figura 7: ***Fórmula per a calcular la distància***

On n és el nombre de colors que composen la imatge, pos(B_R^k) fa referència a la posició del bit amb valor 1 i que respon al color k, la R serveix per diferència si es la imatge de consulta Q o de la imatge que estem buscant la seva signatura binària, en altres paraules la imatge a processar.

2.1.4.- BIC

És un algorisme proposat per Stehling et. al l'any 2002. Consisteix en un algorisme de tres components, un primer component que classifica els pixels de la imatge com a contorn o interior, un segon component que consisteix en una funció logarítmica per a comparar distàncies entre histogrames i finalment un tercer, que consisteix en una forma compacta per representar les propietats.

El primer component assumeix que la imatge estarà sobre un espai RGB de 64 colors, i que un pixel sols utilitza els 4 pixels adjacents. Realitza la classificació per contorn assumint que serà contorn si un dels quatre pixels veïns és d'un color diferent. Si els quatre pixels veïns contenen el mateix color aquest pixel serà interior. Un cop ha realitzat aquesta classificació calcula els histogrames per a cada una de les categories.

La mesura de distància es realitza utilitzant una mesura de distancia logarítmica per intentar reduir la diferència entre distàncies molt grans i distàncies molt petites. La funció en qüestió és la que s'aprecia en la figura 8.

$$dLog(q,d) = \sum_{i=0}^{i<M} \left| f(q[i]) - f(d[i]) \right|$$

Figura 8: ***Funció per a calcular la distància logarítmica***

Un cop calculats els histogrames, cal guardar els valors de cada barra. La forma de guardar-lo no consisteix en guardar el valor normalitzat, sinó que guarden el valor equivalent en l'escala logarítmica. Aquest canvi consisteix en assignar valor 0, si el punt que estem calculant és el 0, valor 1 si esta entre 0 i 1, o $\log_2 +1$ en la resta de casos.

2.2.- Textura

La textura és una propietat de les superfícies que descriu patrons visuals seguint propietats d'homogeneïtat. Aquesta propietat conté informació important sobre l'organització estructural de la superfície. També descriu la relació de la superfície amb l'entorn.

Pel que fa a imatges, és una de les característiques descriptives més importants, esta caracteritzada per la distribució dels nivells de gris en els veïns.

A l'hora de descriure la textura ens trobem bàsicament amb tres mètodes. Un primer mètode conegut com tècniques estadístiques, aquest mètode pretén caracteritzar les textures utilitzant les propietats espacials dels nivells de grisos de la imatge. Dos algoritmes d'aquest grup serien la matriu de co-ocurrencia de la superfície i el que es coneix com transformació Wavelet de la superfície. El segon mètode que s'engloba dins la caracterització per tècniques estructurals, parteix de la idea que la textura esta composada per primitives simples conegudes com a "textels". La textura de la imatge serà una composició d'aquests elements simples. Finalment la tercera família englobada dins les tècniques espectrals, es basa en escènica en les propietats de l'espectre de Fourier i descriu la periodicitat global dels nivells de gris en una superfície, s'aconsegueix identificant els pics d'energia dins l'espectre de Fourier.

En sistemes CBIR el més comú es utilitzar la matriu de co-ocurrencia, la Tamura Texture o les transformacions basades en wavelets.

2.2.1.- Matriu de Co-ocurrencia

Representa les característiques de textura explorant la dependència espacial del nivell de grisos d'una imatge, fou proposada originalment per R.M.Haralick. L'algorisme per a calcular aquesta matriu és el següent:

Donat un operador de posició P(i,j) i sigui A una matriu de tamany n x n els elements de la qual son el nombre de vegades en que una determinada intensitat de gris g[i] es repeteix en la posició especificada per P i relativa als punts amb nivell de gris g[j].

Sigui C una matriu de tamany n x n obtinguda dividint la matriu A pel nombre total de punts. Cada posició C[i,j] d'aquesta matriu representarà la probabilitat que un parell de punts tinguin el valor g[i] i g[j] respectivament. Un exemple es pot observar en la figura 9.

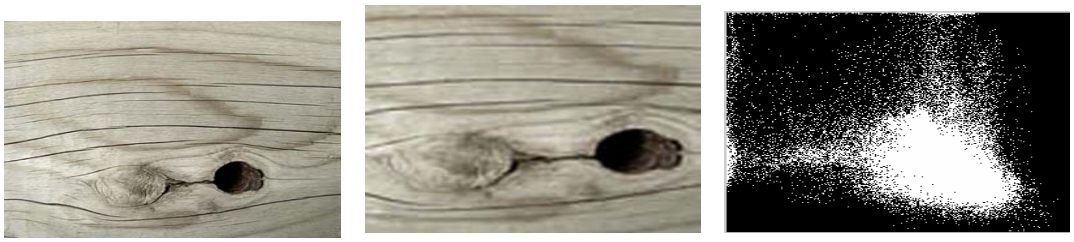


Figura 9: ***Representació gràfica de la matriu de co-ocurrencia de dues imatges***

Un cop tenim la matriu construïda podem obtenir estadístiques de la textura de la imatge com per exemple el moment angular, el contrast, la correlació, la variància, la suma mitjana, la suma variància ...

2.2.2.- Textura Tamura

Tamura explora la representació de la textura utilitzant aproximacions computacionals de les característiques principals de la textura. Aquest sistema proposa sis caracteritzadors de textura la granularitat, el contrast, la direcció, la línia de semblança, la regularitat i la rugositat. En general les tres primeres característiques són les més diferenciadores pels humans. [17]

La granularitat, en aquest cas, consisteix en el tamany mitjà de les regions que tenen la mateixa intensitat. Per extreure aquesta característica comencem calculant sis mitjanes per a cada pixel, cada una d'aquestes mitjances s'obté variant el tamany de la finestra. La finestra és quadrada, de tamany 2^k x 2^k i centrada en el pixel que estem buscant, variant k entre 0 i 5, obtenim les 6 mesures que desitgem.
Un cop obtingues, novament per a cada pixel, calculem la diferencia absoluta entre la suma de la fila i la columna que no es sobreposen. Seguidament busquem el valor de la finestra pel qual aquesta diferència és màxima i el guardem en la posició S(x,y). La característica de granularitat vindrà definida per la mitjana de la matriu S(x,y).

El contrast és la mesura del patró de textura, s'aproxima utilitzant blocs amb variació d'intensitats de blanc i negre.

La direcció és la mesura de la direcció dels valors en la escala de grisos dins la imatge, aquesta característica s'extreu utilitzant la distribució de freqüència aproximada dels contorns locals enfront dels angles que segueixen. Aquests dos valors es calculen utilitzant el filtre de Sobel (detecció de contorns). [18]

En general aquesta informació es guarda en un histograma de tres dimensions, el qual es quantificat en M = 6 x 8 x 8 = 384 barres. Per a poder utilitzar aquesta informació de forma útil cal que les imatges de les quals extraiem aquesta informació estiguin escalades a 256x256. Un cop tenim els histogrames, requerim d'una funció que els pugui comparar, en aquests tipus de situacions s'acostuma a utilitzar la fórmula de la divergència de Jensen-Shannon. [19]

2.2.3.- Transformacions Wavelet

Les textures poden ser modelades com patrons gairebé periòdics en la representació espai / freqüència. Aquestes transformacions converteixen la imatge en una representació multi escala utilitzant ambdues característiques. Aquest mètode ens permet un anàlisis multi escala amb un cost computacional relativament baix. [11]

Si Fourier utilitzar funcions sinus per a representar les senyals, aquí s'utilitzen funcions conegudes com Wavelets. Aquestes funcions són finites en el temps i el valor mitjà d'aquestes funcions és 0.

Existeix una gran varietat de Wavelets, com per exemple Coiflet, Morlet, Mexican Hat, Haar i Daubechies. Haar és el més simple i el que es recomana no utilitzar en aplicacions CBIR, per contra, els coeficients de Daubechies amb les seves estructures fractals són vitals pels sistemes CBIR.

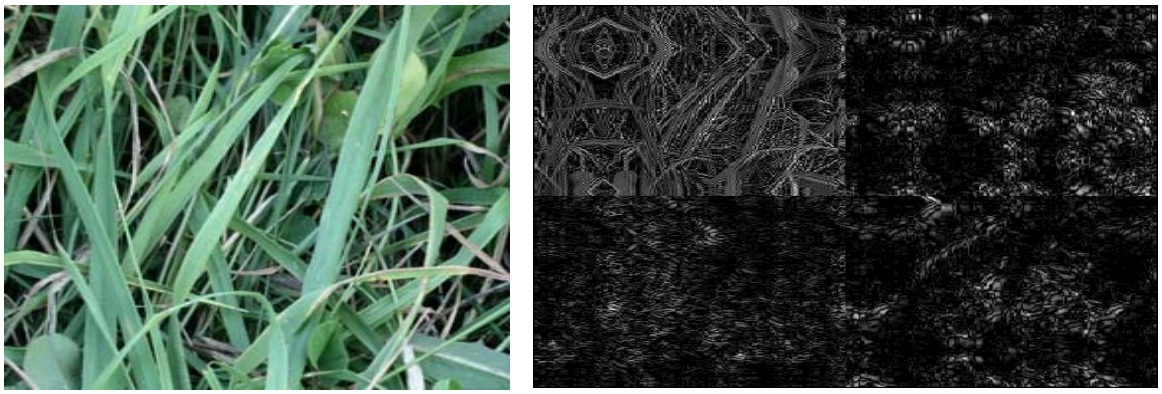


Figura 10: ***Descomposició en coeficient de Daubechies d'una imatge***

2.3.- Forma

La forma pot ser definida com la característica de configuració d'un objecte sobre una superfície, en altres paraules el separador de regió o contorn. A l'hora de representar la forma ens trobem amb dues categories.[13]

Una primera categoria coneguda com a basada en el contorn, com el seu nom indica aquesta categoria utilitza el contorn de la forma, s'aconsegueix per exemple buscant els pixels frontera. La segona categoria coneguda com basada en regió, utilitza tots els pixels de la regió. Com a exemples de la primera categoria podríem trobar els models poligonals o les corbes splines, per la segona família els superquadrics o els esquelets de Blum en serien els exemples més significatius.

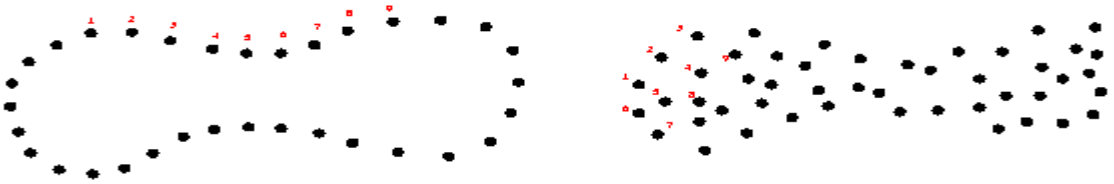


Figura 11: ***Interpretació gràfica de forma basada en contorn i basada en regió.***

Les proves experimentals han demostrat que les millors representacions son els descriptors de Fourier per la segona categoria i els moments invariants per la primera. La idea dels descriptor de Fourier es basa en utilitzar la transformada de Fourier com indicador de contorn, com es veurà en el punt 3.3.1. Els moments invariants es basen en utilitzar els moments de la regió, els quals son invariants a transformació, com a mesura de forma.

3.- Test d'algorismes

3.1.- Funcions de distancia

Tots els algorismes per detecció de característica en última instància requereixen trobar quina de les imatges es la més propera a la sol·licitada, per aquest motiu escollir una bona funció de distància es un punt important.

Figura 11: ***Imatge de petició per els tests***

La figura 11 serà la imatge de consulta que utilitzarem com a imatge de petició en les diferents implementacions, el que ens permetrà comparar els diferents mètodes de distància de forma objectiva.

3.1.1.- Distancia de Mahalanobis

És un tipus de distància que es caracteritza per determinar una mesura de semblança entre dues variables aleatòries amb dimensions múltiples, com a punt fort destacar que aquest mètode te en compte la correlació entre les variables aleatòries. La fórmula és la que s'observa en la figura 12. On la variable del denominador és la variància de i.

$$d_2(\vec{x_1}\vec{x_2}) = \sqrt{\left(\frac{(x_{11} - x_{12})}{\sigma_1}\right)^2 + \left(\frac{(x_{21} - x_{22})}{\sigma_2}\right)^2}$$

Figura 12: ***Distància de Mahalanobis.***

Els resultats d'aplicar aquesta mesura de distància amb unes característiques ja extretes de la imatge són els que es poden observar en la figura 13.

distància = 0.93 distància = 1.01 distància = 1.34

Figura 13: ***Resultats i distància al aplicar l'algorisme de Mahalanobis***

3.1.2.- Distància euclidiana

És la distància més ràpida de calcular, ve definida per la fórmula expressada en la figura 14.

$$d(p_1, p_2) = \sqrt{pow(x_2 - x_1, 2) + pow(y_2 - y_1, 2)}$$

Figura 14: ***Fórmula per la distància euclidiana***

Els resultats d'aplicar aquesta mesura de distància amb unes característiques ja extretes de la imatge són els que podem observar en la figura 15.

distància = 1.68 distància = 1.83 distància = 2.42

Figura 15: ***Resultats i distàncies obtingudes mitjançant la distància euclidiana***.

3.1.3.- Distància de Manhattan

Podem definir la distància de Manhattan com la suma de les longituds de les projeccions dels segments entre dos punts. La fórmula en aquest cas és la que s'indica en la figura 16.

$$|x_1 - x_2| + |y_1 - y_2| .$$

Figura 16: ***Funció per a calcular la distància de Manhattan.***

Els resultats d'aplicar aquesta mesura de distància amb unes característiques ja extretes de la imatge són els que podem observar en la figura 17.

distància = 3.9245 distància = 4.037 distància = 4.3071

Figura 17: ***Resultats* i *distància* utilitzant el mètode de *Manhattan*.**

3.1.4.- Conclusions

Cal dir que en aquest apartat s'han mostrat tres mesures de distància diferents, cal dir però, que n'existeixen moltes més, probablement algunes donarien uns resultats més exactes, però pel treball que s'ha realitzat la distància de Mahalanobis ja compleix amb les condicions que es volien a priori. Una fórmula per a calcular distàncies que fos ràpida i dones uns resultats acceptables.

En els tres casos es pot veure que les formules responen de forma semblant, però tant la distància Euclidiana com la distància de Manhattan ens allunyen molt les imatges semblants, la primera en aquest cas.

3.2.- Detecció de color

Com es mencionava en la introducció el color és una de les característiques més diferenciadores d'una imatge pels humans. Per aquest motiu no podia faltar un apartat dedicat a aquesta característica. La figura 18 serà la que s'utilitzarà en aquest cas per realitzar les proves.

Figura 18: ***Imatge* de *Consulta* pels algorismes de color.**

3.2.1.- Mètode de l'histograma

S'utilitzaran histogrames locals, tal i com es suggeria en la introducció, ara bé caldrà convertir l'espai de color RGB en espai HSV, és una de les condicions que exigeix Matlab per a poder utilitzar les seves funcions. Utilitzarem la fórmula de la distància quadràtica per a buscar la similitud, ja que és la més senzilla d'implementar i la més usada actualment.

3.2.1.1.- Distància quadràtica

Utilitzant histogrames ens cal definir una nova mesura de distància, les formules anteriors ja no ens serveixen, ja que els histogrames no treballen en vectors, sinó amb un histograma. La mesura de distància que utilitzarem en aquest cas és la que es coneix com la fórmula de la distància quadràtica. Consisteix en aplicar la fórmula de la figura 19.

$$d^2(Q,I) = (H_Q - H_I)^t A(H_Q - H_I)$$

Figura 19: ***Fórmula per a calcular la Distància quadràtica***

On A = $[a_{i,j}]$ és la matriu de similitud, fa referència a la semblança entre els colors corresponents als índexs "i" i "j". (Hq - Hi) és la diferencia entre dos histogrames. És en realitat un vector d'una fila i tantes columnes com "barres" tingui l'histograma.

Com veurem més endavant aquesta funció no dona uns resultats massa bons, ja que si l'histograma presenta alguna barra amb un valor molt alt, alta concentració de pixels, té una gran influencia negativa en la distancia d'aquesta imatge respecte la resta

3.2.1.2.- Vectors de coherència de colors

Aquest algorisme esta pensat per pal·liar el problema de la falta d'informació espacial dels histogrames globals[6]. Aquest mètode defineix la coherència de color com el grau en que els pixels d'un determinat color pertanyen a regions lo suficientment grans de colors similars. Aquest mètode es basa en comparar les imatges en base a la seva classificació de pixels coherents i incoherents.

L'algorisme és el següent:

1- Distorsionar lleugerament la imatge **substituint el valor dels pixels pel valor mitjà dels seus veïns**. Amb aquesta distorsió eliminem petites diferencies entre pixels adjacents.
2- **Reduir l'espai de colors** existents en la imatge, s'aconsegueix agrupant totes les tonalitats del mateix color en una única barra del diagrama.
3- **Calcular les regions connectades**, bàsicament es tracta de buscar taques del mateix color dins la imatge. Aquests blocs no poden tenir un tamany aleatori, acostuma a ser un tamany fix i el mínim s'acostuma a fixar amb un valor equivalent a 1% del tamany de la imatge. Si una imatge te 10000 pixels, una regió connectada serà aquella que tingui com a mínim 100 pixels contigus del mateix color.
4- Per cada pixel de la imatge **determinar si** aquest **és coherent** o no.
5- Per a cada color diferent en l'histograma es **calcula la parella (A_j , B_j)** on A representa el nombre total de pixels coherents del color j i B el nombre de pixels incoherents. Per tant l'histograma queda expressat com indica la figura 20.

$$H = \left((\alpha_1,\ \beta_1),\ (\alpha_2,\ \beta_2),\ ...,\ (\alpha_n,\ \beta_n)\right)$$

Figura 20: ***Expressió de l'histograma, utilitzant el mètode de vectors.***

6- Un cop tenim dos histogrames calculats:

$$G = \left((\alpha_1,\ \beta_1),\ (\alpha_2,\ \beta_2),\ ...,\ (\alpha_n,\ \beta_n)\right)$$

$$G' = \left((\alpha'_1,\ \beta'_1),\ (\alpha'_2,\ \beta'_2),\ ...,\ (\alpha'_n,\ \beta'_n)\right)$$

Figura 20: ***Dos histogrames expressats en notació vectorial.***

Calculem la distància entre ells mitjançant la funció expressada en la figura 21.

$$d(G,G') = \sum_{j=1}^{n} \left|\alpha_j - \alpha'_j\right| + \left|\beta_j - \beta'_j\right|$$

Figura 21: ***Fórmula de la distància per vectors de coherència.***

3.2.1.3.- Resultats

Els resultats obtinguts amb el mètode de l'histograma s'observen en la figura 22.

Figura 22: ***Resultats obtinguts utilitzant el mètode d'histogrames locals.***

3.2.2.- Segmentació de la imatge

La segmentació de la imatge és un procés molt costos computacionalment, i en aquest cas no necessitem tenir tota la imatge segmentada, ens interessa únicament els nuclis de segmentació i el nombre d'elements que conté. Per aquest motiu ens quedarem amb la primera part de l'algorisme de segmentació. La part que busca els nuclis de segmentació. Utilitzarem l'algorisme k-means per aquest propòsit.

L'algorisme k-means consisteix bàsicament en dividir els objectes, pixels en aquest cas, en diferents agrupacions anomenades clusters.

L'algorisme comença realitzant particions dels objectes en k conjunts, de forma aleatòria o seguint alguna heurística (segons la implementació). Seguidament calcula el punt mig, conegut com a centroide de cada set. Un cop ha realitzat aquest càlcul recalcula de nou la distribució dels pixels en funció de la distància als centroides, el centroide més proper es queda amb aquest pixel. Novament es recalculen els centroides de la nova agrupació i es repeteix aquest procés d'associació dels pixels i de càlcul dels centres fins que el mètode convergeix, en altres paraules, fins que els centroides no canvien de valor.

Els resultats obtinguts amb aquest mètode es poden observar en la figura 23.

Figura 23: ***Resultats utilitzant l'algorisme K-means***.

3.2.3.- Conclusions

L'exemple de prova parla per si sol, utilitzant l'algorisme de k-means, els resultats obtinguts són semblants al que "a priori" s'esperava, el mètode de l'histograma en aquest cas difereix molt d'una solució acceptable del problema. Cal dir que sols es mostra un exemple, però en les diverses proves realitzades, l'algorisme de k-means proporciona una millor solució en tots els casos.

Els resultats no són sorprenents, ja que el mètode de l'histograma no ens aporta cap informació espacial, en altres paraules sols ens indica la quantitat de colors, no la seva localització, per tant imatges completament diferents poden tenir histogrames similars com veurem amb l'exemple de la figura 24.

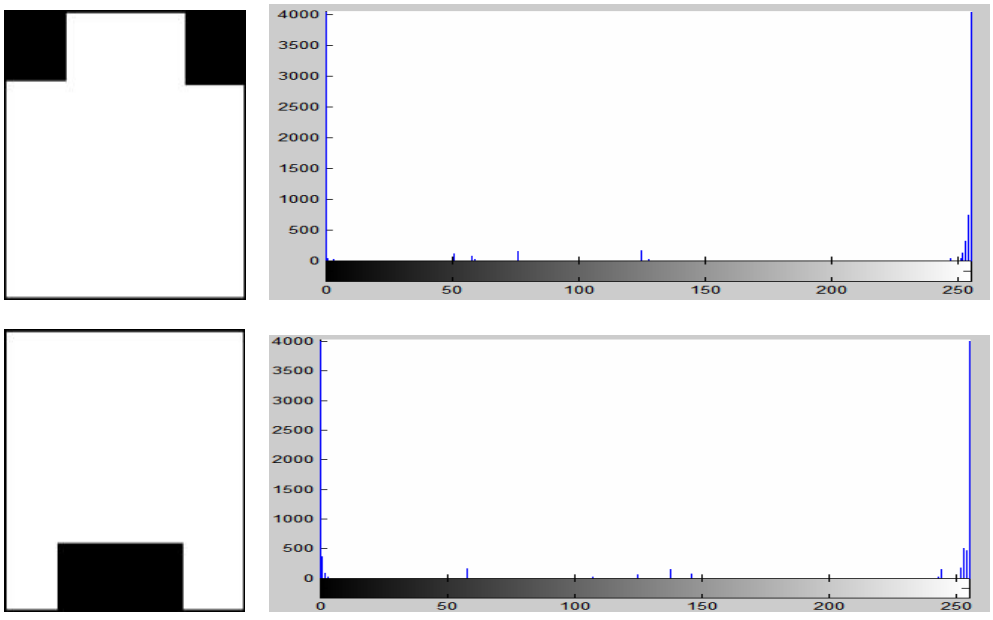


Figura 24: ***Dos imatges completament diferents amb histogrames semblants.***

3.3.- Detector de forma

Segons el conjunt d'imatges la obtenció de bons resultats és pot aconseguir simplement utilitzant característiques de color o característiques de textura. La intenció del projecte era obtenir un mètode general per a realitzar CBIR, per aquest motiu aquest pas no s'ha omès i s'ha realitzat també una extracció de característiques utilitzant la forma de la imatge com a element base. Quan es parlava de forma es deia que existien dos categories, i cada categoria disposava d'un mètode relativament eficient. En aquest punt posarem a prova aquests mètodes utilitzant la imatge mostrada per la figura 25.

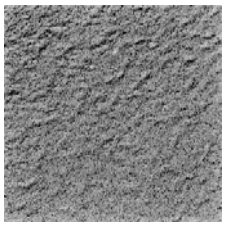


Figura 25: ***Imatge de consulta per a les proves amb algorisme de forma.***

3.3.1.- Coeficients de Fourier

El mètode de Fourier es basa en utilitzar precisament els coeficients obtinguts per la transformada de Fourier com a quantificador de forma de la imatge. En general sols s'acostuma a utilitzar la fase ja que es més representativa de la imatge que la magnitud. La representació gràfica de dues imatges es pot observar en la figura 26.

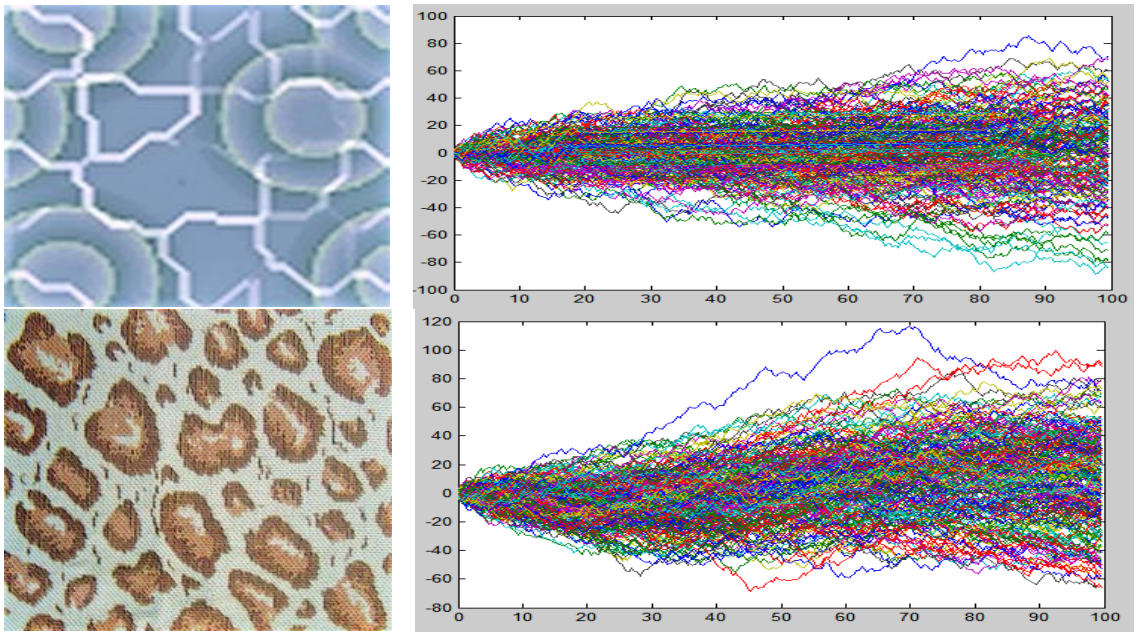


Figura 26: ***Representació de la fase de l'espectre de Fourier de dos imatges.***

Els resultats utilitzant aquest algorisme es poden observar en la figura 27.

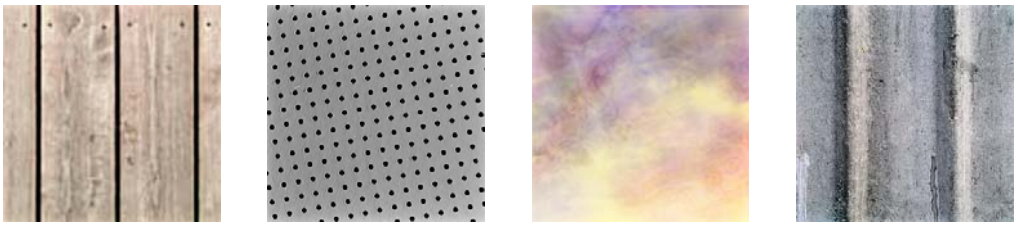


Figura 27: ***Resultats utilitzant els coeficients de Fourier.***

3.3.2.- Detector de contorns

Aquest procediment, com el seu nom indica, consisteix en obtenir les línies separadores de frontera, o contorns, dels objectes dins una imatge. El procediment per a realitzar-lo dependrà de l'algorisme utilitzat. Existeixen diferents famílies:

- **Operadors Gradient**: La forma típica d'establir el gradient d'una imatge és mitjançant el producte de la imatge per dues màscares que representen la magnitud del gradient en dos direccions perpendiculars. Seguint les formules indicades en la figura 28.

$$g_1(m,n) = \langle U, H_1 \rangle_{m,n} \quad g(m,n) = \sqrt{g_1^2(m,n) + g_2^2(m,n)}$$
$$g_2(m,n) = \langle U, H_2 \rangle_{m,n} \quad \theta_g(m,n) = tan^{-1}\frac{g_2(m,n)}{g_1(m,n)}$$

Figura 28: ***Formules de les màscares per a operadors gradient.***

Normalment la magnitud del gradient de la imatge es calcula com:

$$G(x,y) = |G_1(x,y)| + |G_2(x,y)|$$

Un pixel és declarat com a frontera si excedeix un determinat valor llinda, és comú utilitzar un valor llinda en funció de l'histograma acumulat G(m,n) de forma que sols representi entre un 5% i un 10% del total de pixels.

En computació s'utilitzen bàsicament tres mètodes amb aquest operador. El mètode Roberts, el mètode Smoothed també conegut com Prewitt i el Sobel. En la figura 29 es pot observar les màscares utilitzades per cada un dels mètodes.

Operadors gradient	Màscara 1	Màscara 2
a) Roberts	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$	$\begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$
b) Smoothed (o Prewitt)	$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$
c) Sobel	$\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$	$\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$

Figura 29: ***Taula que mostra les màscares que utilitza cada mètode.***

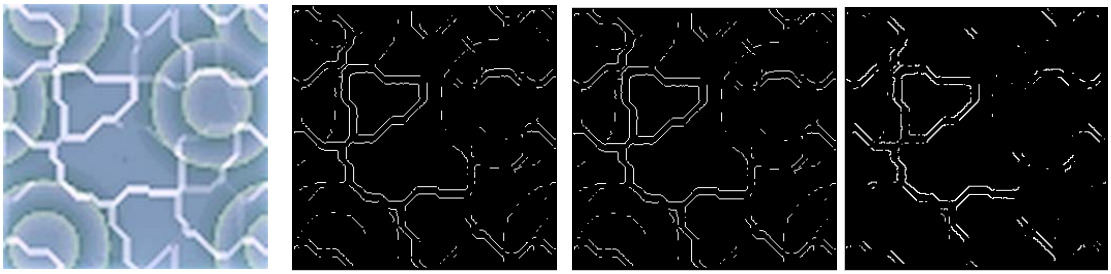


Figura 30: ***Exemples d'aplicació dels mètodes Sobel, Prewitt i Roberts en una imatge.***

Aquest grup treballa bé en imatges on el contorn pateix un canvi molt brusc, dit formalment on la transició de nivells de grisos és molt abrupta, com a avantatge destacar que té un cost computacional molt baix, però els resultats que ens ofereix, són força deficients pel nostre problema.

- **Operadors Laplacians i Creuaments amb zero:** Aquesta família opera amb la segona derivada, el que li permet detectar regions de transicions sense canvi abrupte. La segona derivada presenta un problema, el que es coneix com doble contorn (apareix un màxim i un mínim), per solucionar aquest problema s'utilitza el que es coneix com Creuament amb zero i consisteix en considerar contorn si creua en zero. Cal dir però que no sempre van units en una única implementació, sovint es troben en implementacions diferents.

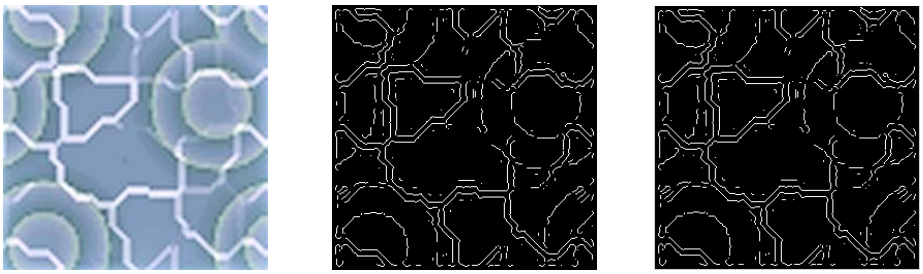


Figura 31: ***Exemples d'aplicació de l'operador Laplacià, i Creuament amb zero.***

Com s'observa aquesta família obté uns resultats bastants acceptables, detecta la majoria de contorns que a simple vista podríem distingir.

- **Algorisme Canny**: Aquest algorisme pretén ser una unió entre tots els detectors de contorns. Primerament filtra el possible soroll que aquesta tingui, seguidament busca el gradient utilitzant una successió de derivades, no sols la primera i la segona. Seguidament elimina aquells punts en els que el nivell de derivació amb el que esta operant no és màxim. El següent pas consisteix en elimina els falsos positius utilitzant dos valors llinda, si el valor és inferior al llindar més baix, aquest punt ja no és frontera, si el punt esta per sobre el llinda més alt, aquest serà un punt frontera. La clau esta en els valors que estan dins l'interval, en aquest cas, mira si els pixels veïns són frontera, si ho són, aleshores aquest punt també serà frontera. La figura 32 mostra un exemple d'ús d'aquest mètode.

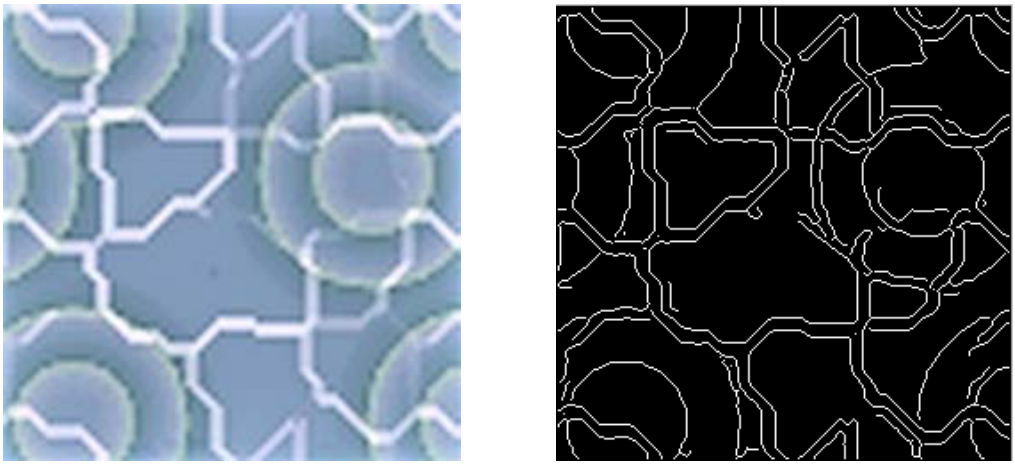


Figura 32: ***Exemple d'aplicació de l'algorisme de Canny.***

Aquest mètode ens proporciona una solució més que notable, per aquest motiu molts estudis consideren aquest mètode com el més adient per a treballar en CBIR i en general per a qualsevol aplicació que requereixi un reconeixement de contorns. Per aquest motiu d'ara endavant s'utilitzarà aquest algorisme.

Els resultats d'aplicar un detector de contorns basat en l'algorisme de Canny es poden observar en la figura 33.

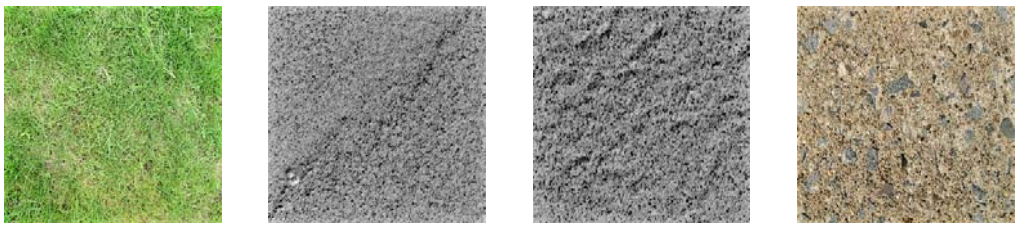


Figura 33: **Resultats d'aplicar l'algorisme de Canny per a detectar contorns.**

3.3.3.- Conclusions

El mètode de Fourier no ens proporciona la sortida esperada, les imatges que retorna disten molt de la imatge que proposàvem com a consulta, el detector de contorns utilitzant l'algorisme de Canny ens retorna unes imatges similars en quan a configuració d'objectes.

3.4.- Detecció de textura

La textura d'una imatge, és la tercera característica que podem obtenir d'una imatge i conjuntament amb el color permet distingir de forma clara una imatge d'una altra. Novament utilitzarem la imatge de la figura 34 per a realitzar el test entre els diferents algorismes disponibles.

Figura 34: ***Imatge de test.***

3.4.1.- Daubechies Wavelets

Consisteix en descompossar les imatges en canals de freqüències. És idoni per textures amb canals de freqüència dominant. La textura de la imatge es descompon en quatre subimatges, cada subimatge es correspon a diferents canals, el canal de freqüències baix- baix, el canal de freqüències alt- alt, i les altres dues son les combinacions baix- alt i alt- baix.[14] En aquest punt es calcula l'energia de cada un dels canals. Un exemple d'ús es pot veure en la figura 35.

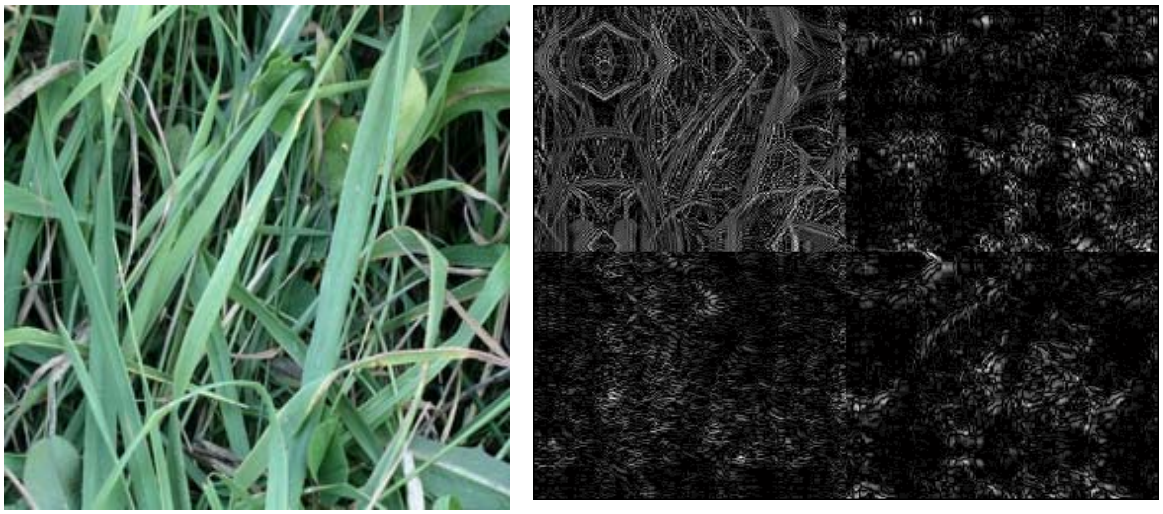


Figura 35: ***Descomposició en coeficients de Daubechie d'una imatge.***

La figura 36 mostra els resultats d'aplicar la transformació en Wavelets utilitzant els coeficients de Daubechie.

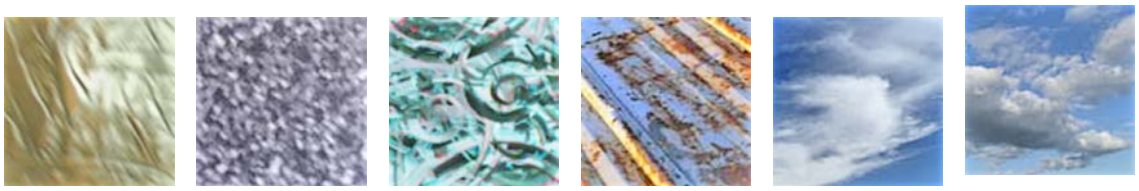


Figura 36: ***Resultats utilitzant els coeficients Daubechie***

3.4.2.- Gradient

Utilitzar el gradient com a detecció de textura consisteix en calcular la magnitud i la fase d'aquest. La mitjana i la variància actuaran com a descriptors de la textura. Amb aquest mètode obtenim un indicador d'uniformitat de la imatge. Es pot observar un exemple en la figura 37.

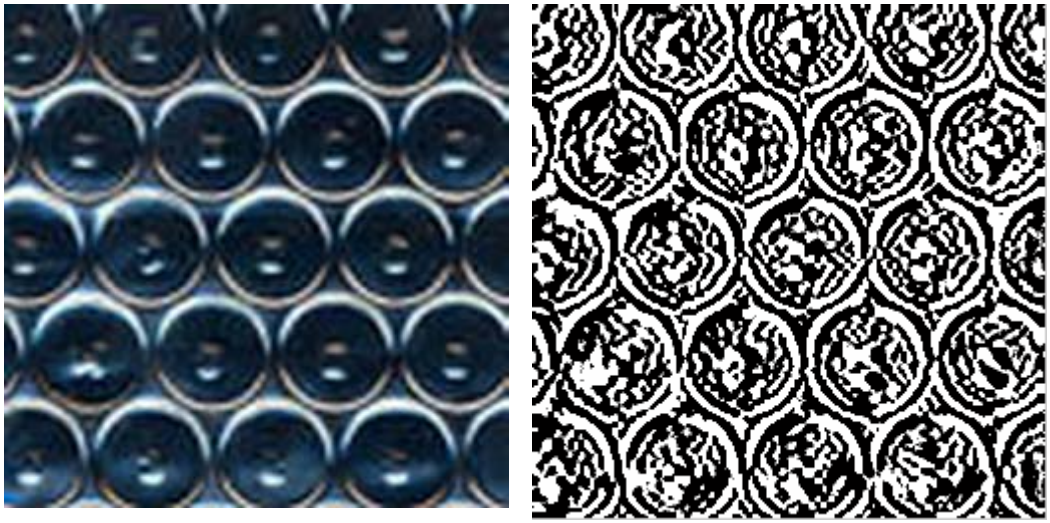


Figura 37: ***Representació gràfica del gradient d'una imatge.***

La figura 38 mostrarà els resultats obtinguts aplicant el mètode del gradient.

Figura 38: ***Resultats aplicant el mètode del gradient.***

3.4.3.- Conclusions

Els dos mètodes retornen resultats similars, potser destacar la posició en que detecta la primera imatge dels núvol el mètode del gradient. Però la segona del núvol que detecta dista bastant, al menys aparentment, de la textura de la imatge de petició. Per aquest motiu i per les recomanacions, d'estudis més extensos, d'utilitzar els coeficients de Daubechies s'opta per aquest algorisme com a part de la solució final.

4.- Solució proposada

La solució proposada es basa en una part prèvia de processament de les imatges que consisteix a realitzar les normalitzacions de la imatge, per tal que es pugui realitzar una comparació objectiva d'elles.

L'extracció de característiques segueix una estructuració en forma d'arbre, on en cada nivell s'efectua un filtre d'imatges utilitzant la característica adient com a discriminant. S'ha optat per aquest mètode per millorar el temps de càlcul i l'eficiència del sistema. Una argumentació d'utilitzar aquest algorisme seria per exemple que si una imatge difereix molt en el color respecte la imatge de petició, no cal realitzar els passos de cerca de textura o de contorn per aquesta imatge, ja que possiblement aquesta imatge no sigui semblant, si més no, els humans no la percebrem com a similar. Tot i que aquesta idea pot plantejar un problema i és el següent, dos imatges semblants on sols canvia el color no serien detectades semblants, però en realitat no és així, l'algorisme de k-means no es basa en quin color és, sinó en la concentració de zones de colors, per tant la detectaria com a semblant sense cap problema.

4.1.- Normalització

La solució proposada, com dèiem s'inicia amb una etapa de normalització. Una normalització de tamany de totes i cada una de la imatge, s'ha optat per treballar amb imatges de tamany 256x256 perquè és un tamany suficient per poder realitzar les proves amb resultats indicatius i perquè el processament d'imatges d'aquest tamany no resulta excessivament costos computacionalment.

La segona consisteix en eliminar les "interferències" causades per les diferents il·luminacions que pot tenir una imatge, aquest procés es coneix com a normalització fotomètrica, en el nostre cas utilitzarem l'algorisme Indane, que es basa en els següents punts:

- Desplaçar la il·luminació amb la transformació de la intensitat del pixel. Primerament convertim la imatge en escala de grisos i seguidament apliquem la fórmula de la figura 39 per a Normalitzar la intensitat.

$$I_n' = \frac{(I_n^{0.24} + (1 - I_n) \cdot 0.5 + I_n^2)}{2}$$

Figura 39: ***Fórmula per a realitzar la normalització d'intensitat.***

- Obtenir la informació de la il·luminació amb la convolució de dos dimensions Gausianes. Aquest pas es pot veure en la figura 40.

$G(x,y) = K \cdot e^{\left(\frac{-(x^2+y^2)}{c^2}\right)}$ on obtenim K de $\iint K \cdot e^{\left(\frac{-(x^2+y^2)}{c^2}\right)} \cdot dxdy = 1$

convolucionar com

$$I'(x, y) = I(x, y) * G(x, y)$$

Figura 40: ***Convolució en dos dimensions Gausianes.***

- Comparar I'(x,y) amb el centre del pixel

$R(x, y) = 255 \cdot I_n'(x, y)^{r(x,y)}$ on obtenim r(x,y) de

$$r(x, y) = I'(x, y)/I(x, y)$$

Figura 41: ***Comparació amb el centre del pixel.***

La il·luminació del centre del pixel augmenta o disminueix depenent de r(x,y), els resultats òptims s'aconsegueixen aplicant la fórmula de la figura 42.

$R(x, y) = \sum_i w_i R_i(x, y)$ on i són les diferents escales utilitzades (5,20,240) i j és el pes del contrast de sortida (1/3).

Figura 42: ***Obtenció de la il·luminació del pixel central.***

- Finalment l'últim pas consisteix en restaurar el color de la imatge aplicant la fórmula de la figura 43.

$R_j(x, y) = R(x, y) \frac{I_j(x, y)}{I(x, y)} \cdot \lambda$ on j són els tres espectres de banda de color (RGB) λ és per ajustar el color RGB.

Figura 43: ***Restauració del color de la imatge.***

Aquest algorisme esta implementat en la funció indane.m la que a la seva vegada crida a la funció luminance_cdf.m que és una optimització per trobar la luminància coneguda com Aindane. Un exemple d'execució es pot veure en la figura 44.

Figura 44: ***Resultats d'aplicar la normalització fotomètrica.***

4.2.- Construcció de la base de dades

Inicialment es crea un arxiu que conté la ubicació de totes les imatges que ens interessa que formin part de la base de dades. Aquest pas és realitza mitjançant la funció afegir (consultar el punt 9). Aquesta funció genera un arxiu, dadesimatges.txt que té el següent contingut:

C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im001.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im002.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im003.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im004.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im005.bmp

4.3.- Extracció de característiques de color

És la primera part de la funció buscar_na.m, el primer pas que realitza consisteix en buscar els centroides i el seu nombre de pixels mitjançant l'algorisme k-means. Seguidament llegeix l'arxiu dadesimatges.txt i per cada imatge que contingui aquesta llista li aplica també l'algorisme k-means. Arribat en aquest punt es calcula la distància entre vectors utilitzant la distància de Mahalanobis i es guarda en un vector, també s'emmagatzema el nom de la imatge que estem processant. Un cop s'ha obtingut les característiques de totes les imatges s'han d'ordenar els vectors de menor a major, en altres paraules, els primers elements seran aquells que és trobin a una distància inferior respecte la imatge original. Un cop ordenats ens quedem amb una vuitena part del total d'imatges, s'utilitzen tantes imatges perquè es prima la detecció correcta, i ens interessa

que els nivells següents disposin de suficients imatges per a poder realitzar la cerca, encara que contingui molts falsos positius. Aquests resultats es guarden en un arxiu anomenat color.txt. que té el següent contingut:

C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im064.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im448.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im065.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im438.bmp
C:\Documents and Settings\Crash\Escritorio\coses\projecte\imatge\im375.bmp

4.4.- Extracció de característiques de contorn

A diferència del pas anterior, en aquest cas, no es calcula primerament els contorns de la imatge de cerca, sinó que calculem a cada pas la diferència entre els contorns amb cada imatge de les contingudes dins de l'arxiu color.txt. Per a realitzar la detecció de contorns s'utilitza el detector de Canny.

A l'hora de buscar els contorns no es busquen mirant el nombre de pixels amb valor 0 (blanc), el que es realitza és una divisió de la imatge en blocs de vuit per vuit i busquem els pixels blancs d'aquesta zona; amb aquest mètode s'aconsegueix un vector, més formalment, una aproximació local de contorns el que ens proporcionarà uns resultats més precisos.

Un cop calculats els contorns locals es busca la distància utilitzant novament Mahalanobis, i de nou és guarda el vector amb el valor i el nom de les imatges, que posteriorment s'ordenaran i es guardarà una setzena part del total d'imatges en l'arxiu contorn.txt.

4.5.- Extracció de característiques de textura

Per aquest pas novament tornem al mètode inicial, calcular primerament les característiques de textura de la imatge inicial. El càlcul es realitza mitjançant la funció obtainEnergies.m. S'ha optat per utilitzar el que es coneix com arbre Wavelet, que consisteix en calcular diferents nivells de Wavelet. Aquest pas consisteix en realitzar un seguit d'iteracions, 5 en el nostre cas. Aquesta millora consisteix en primerament calcular la descomposició en coeficients de Daubechies de la imatge original, i un cop

calculada anem obtenint els coeficients però utilitzant com a imatge, la imatge de baixes de freqüències calculades en el punt anterior. Una execució es pot veure en la figura 45.

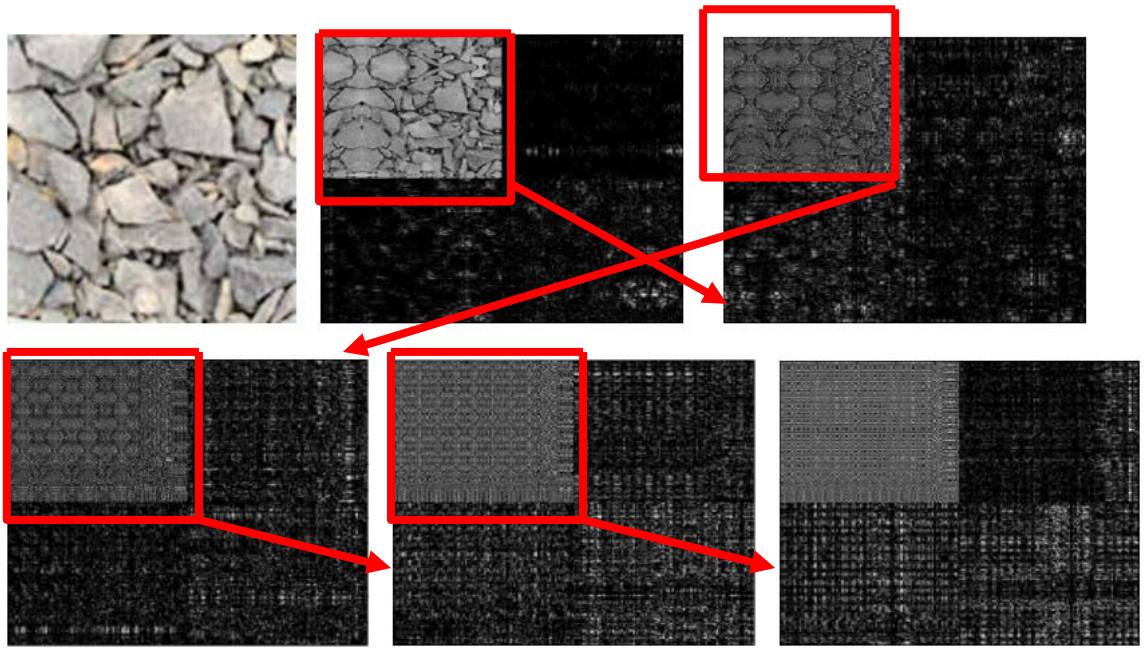


Figura 45: ***Exemple d'execució de l'arbre Wavelet.***

Els coeficients de Daubechies s'obtenen mitjançant la funció decompose.m. En el nostre cas s'utilitza els coeficients de Daubechies d'ordre 45, aquest valor fa referència al nombre de coeficients que s'utilitzaran. A la pràctica aquest valor limita l'aproximació polinomial, si utilitzem valors petits, 2,4 sols ens permetrà aproximar a polinomis d'ordre 1,2 respectivament, en el nostre cas, aproxima la imatge en polinomis d'ordre 22.

En cada iteració guardem els nivells d'energia de les quatre subimatges en un vector, seguidament les ordenem de forma descendent i retornem el nombre que ens interessi, per aquest cas s'ha demostrat que amb 6 és més que suficient.

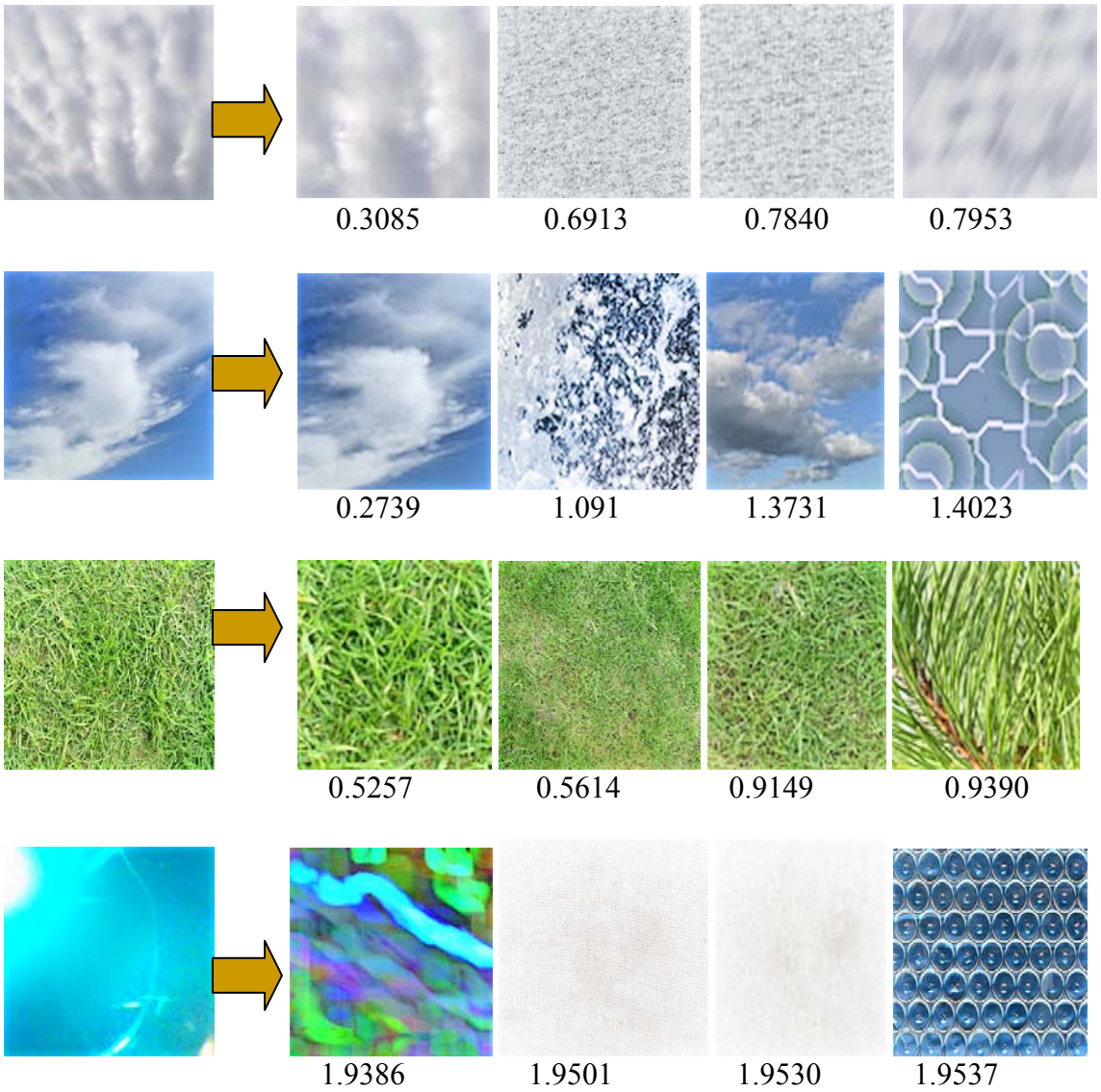
Un cop tenim les 6 energies dominants per a cada imatge, la comparem amb les energies de la imatge de petició, és calcula la distancia entre l'energia de les imatges utilitzant la distancia de Mahalanobis, es guarda en un vector, s'ordenen i guardem una trenta-dossena part, aquestes últimes imatges, seran les imatges finals que retornarà el programa.

5.- Resultats

5.1.- Aplicacions comercials

5.1.1.- CBIRUS

Consisteix en una aplicació comercial distribuïda per l'empresa CIBIRUS, aquesta aplicació utilitza per indexar les imatges el vector de coherència de colors, els resultats de la seva execució són força sorprenents, sobretot tenint en compte que sols utilitzar el color com a característica de la imatge, tot i que presenta un petit problema d'execució o BUG amb imatges d'escala de grisos, no dona cap error, simplement es queda bloquejat. Cal dir que les imatges passades a aquesta aplicació han estat normalitzades utilitzant la transformació Aindane. Alguns exemples d'execució es mostren la figura 46.



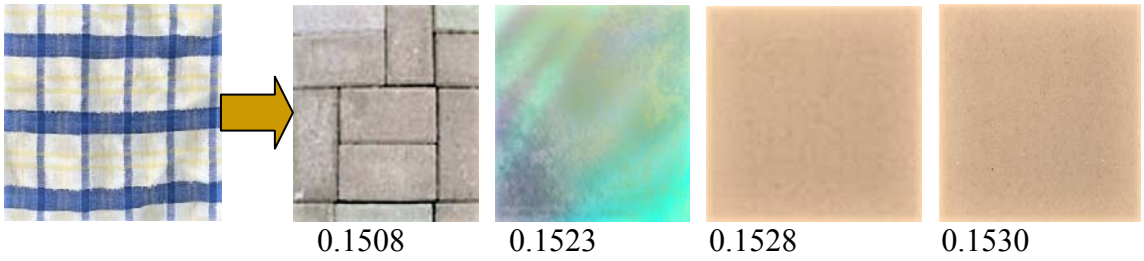


Figura 46: ***Diverses execucions amb els seus resultats de l'aplicació CBIRUS.***

5.2.- Solució Proposada

Un cop executada l'aplicació s'ha comparat manualment si els resultats eren com haurien de ser, en altres paraules, si la percepció humana de la semblança era la que mostrava el programa. Cal dir que no s'ha realitzat la comprovació manual de totes les imatges que contenia la nostra base de dades, s'han realitzat les comprovacions manuals utilitzant un 25 % de la base de dades. S'han anotat manualment els resultats que haurien de sortir de les 116 primeres imatges que representen el 25% que es mencionava. Posteriorment s'han comparat els resultats anotats manualment amb els resultats obtinguts pel programa, aquesta comparació és mostra en la figura 47.

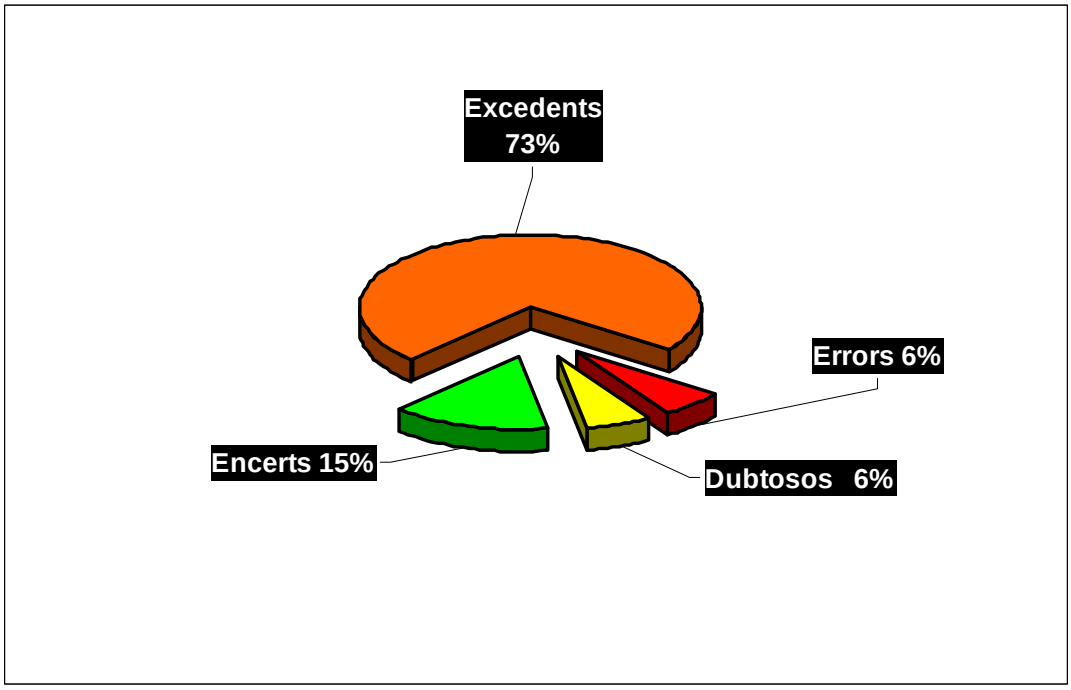


Figura 47: **Resultats detallat de l'execució del 25% de les imatges.**

De la figura 47 cal definir que representa cada grup, el grup dels encerts i el grup dels errors són els que resulten més clars. El grup dels encerts representa les imatges que s'esperaven a priori i que el mètode ha retornat, per contra els errors fan referència a imatges que esperàvem però que no ha obtingut. El grup dels excedents equivaldrien a falsos positius, però no és del tot correcte, recordar que exigíem un determinat nombre de resultats, el més probable que algunes imatges no tinguin aquesta quantitat d'imatges. Finalment el grup de dubtosos són imatges que retorna el programa que a priori no es consideraven semblants, però que si es comparaven amb atenció era difícil establir un veredicte. La figura 48 n'és un exemple.

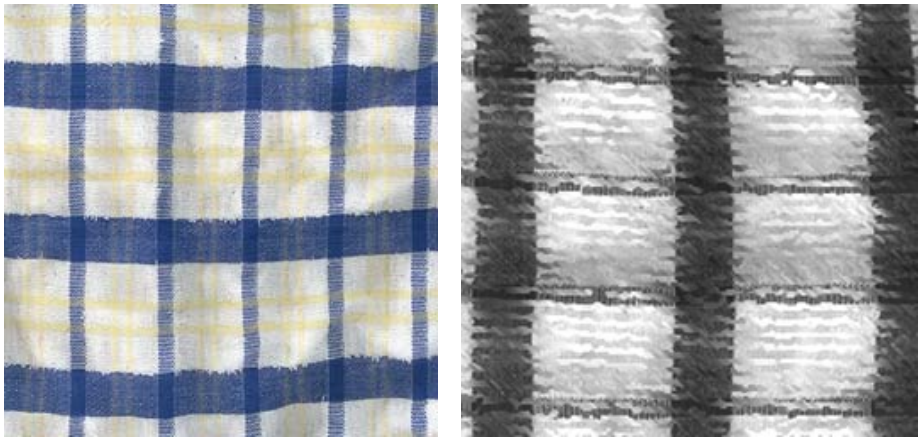


Figura 48: **Exemple d'una comparació dubtosa.**

Diferenciant únicament entre encerts i errors, obtenim els resultats que es mostren en la figura 49.

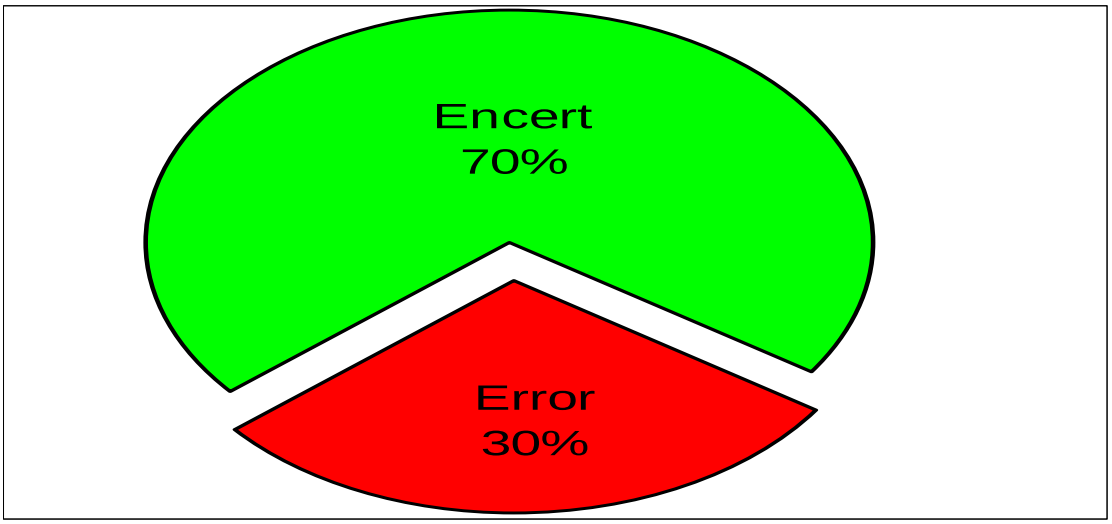


Figura 49: **Resultats de l'execució del 25% de les imatges.**

Un resultat interessant és el que s'observa en la figura 50 i es correspon amb la detecció de la imatge esperada, inicialment es comentava que per a cada imatge existia almenys una que era gairebé idèntica, doncs bé, aquest gràfic mostra el percentatge de detecció d'aquesta imatge.

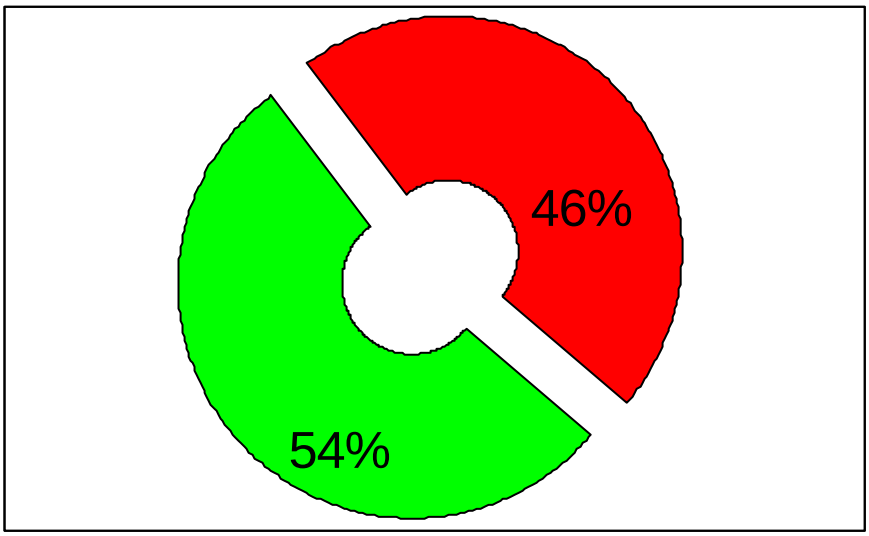


Figura 50: **Resultats respecte la primera imatge esperada del 25% de les imatges.**

Cal dir que la majoria d'aquestes imatges consisteixen en ampliacions d'una zona concreta, o a afegir cert soroll a la imatge, el qual n'alterà completament les característiques que estem buscant.

Per poder realitzar una correcta interpretació dels resultats cal conèixer la desviació i la mitjana de les distàncies que s'obtenen de la prova tots contra tots, aquesta prova consisteix en passar com a imatge de consulta totes les imatges.

La primera parella de resultats consisteix en les distàncies obtingudes durant l'etapa d'extracció de característiques de color, les gràfiques es mostren en la figura 51.

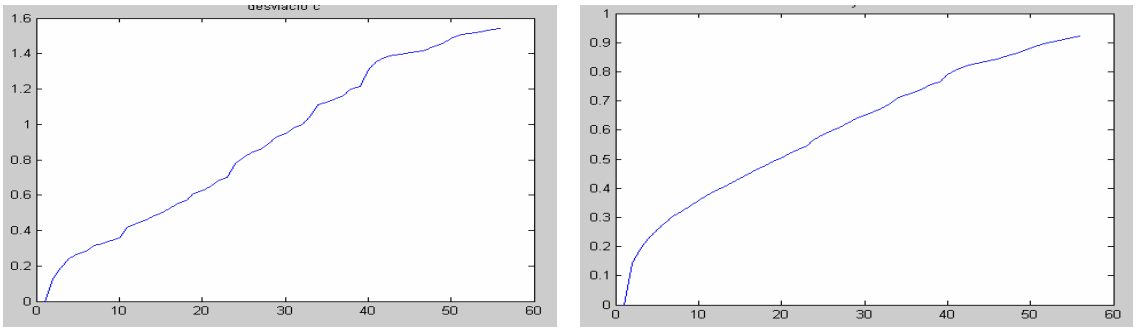


Figura 51: **Gràfica de desviació i gràfica de mitjana de les distàncies de color**

La interpretació d'aquestes gràfiques és trivial, la posició dins l'eix x significa l'ordre que ocupa la imatge trobada. A mode exemple podem interpretar les dues primeres posicions. El primer element ens ve a dir que té una distància mitjana de 0 amb una desviació de 0, aquest resultat és normal, ja que la primera imatge que trobem sempre és ella mateixa. El següent valor ens ve a dir que la segona imatge que troba, està a una distància mitjana de 0.1414 amb una desviació de 0.1218 en les 448 imatges. I d'aquests forma s'interpreten tots els resultats.

Un cop explicat el mètode per interpretar aquestes gràfiques es mostraran les gràfiques per a les dues etapes restants, l'etapa de contorn figura 52 i l'etapa de textura figura 53.

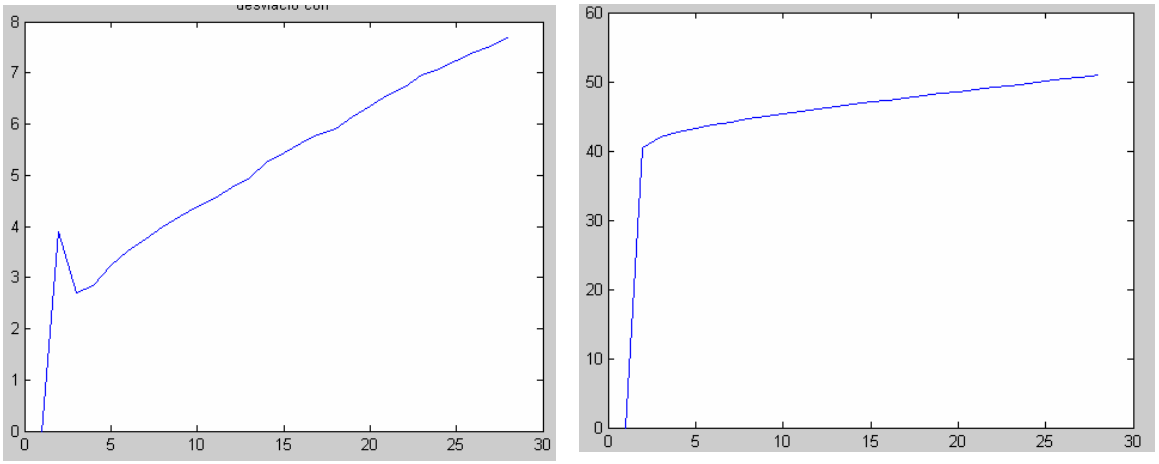


Figura 52: **Gràfica de <u>desviació</u> i gràfica de <u>mitjana</u> de les distàncies de <u>contorn</u>**

Novament es proporcionarà la interpretació dels dos primers valors, el primer resultat obtingut es troba a distància 0 amb una desviació també de 0, pel mateix motiu que es comentava anteriorment, és la mateixa imatge. La segona imatge semblant que troba, està ubicada a distància 40,52 amb una variació de 3.902. Aquesta separació tan gran també es normal, ja que en l'etapa de detecció de contorn buscàvem la diferència de píxels entre finestres de 8x8. Si les imatges tenen tamany 256x256, ens dona que disposem de 1024 finestres. Si s'interpreta la distància com el nombre de finestres diferents que tenen les imatges, aquest valor no és tan desorbitat, un segon argument és calcular el percentatge de diferència, que per aquest cas es troba en 3.9063 %, el que reafirma, novament, que és una bona candidata per ocupar la segona posició.

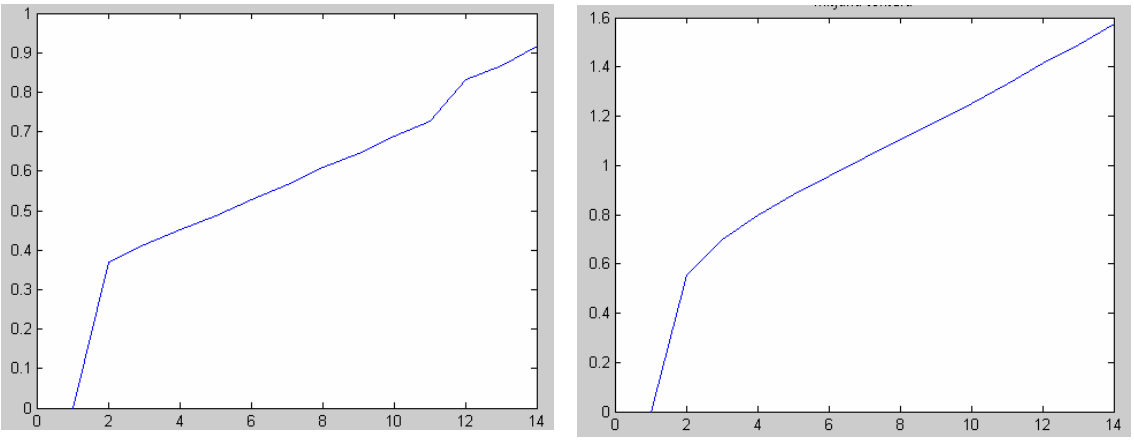
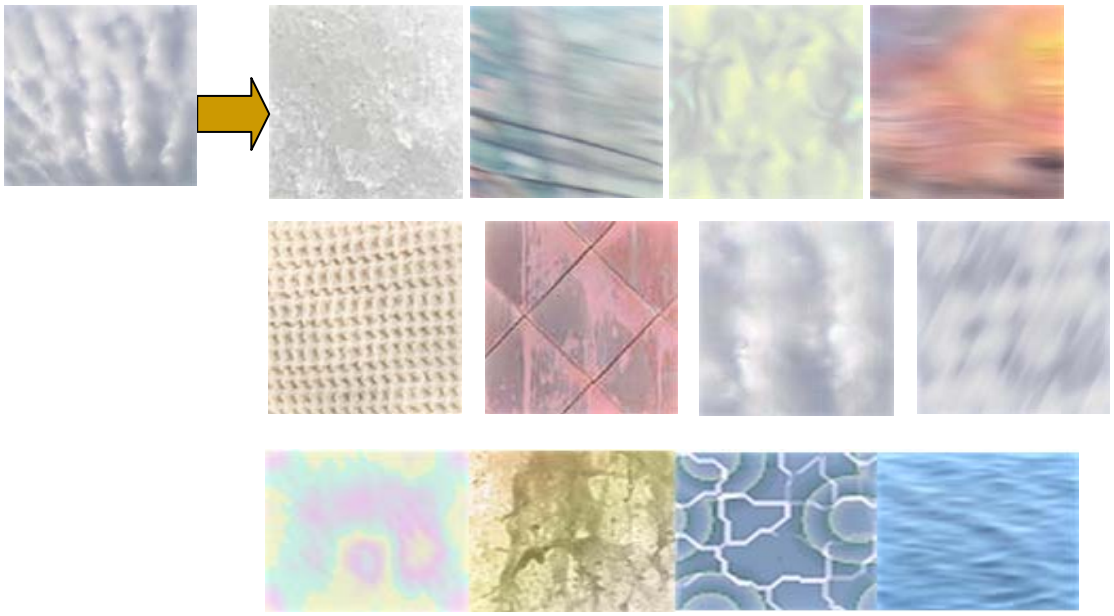


Figura 53: **Gràfica de desviació i gràfica de mitjana de les distàncies de textura**

Novament la primera imatge que troba és la pròpia imatge de consulta, aquesta es troba també a distància 0 i amb una variació de 0, aquest resultat també ens serveix com a indicatiu de correcte funcionament de l'arbre d'execució, en cas contrari no detectaríem en l'últim pas la imatge de consulta com la més propera, i que és idèntica. La segona imatge més semblant en textura es troba a distància 0.55554 amb una variació de 0.36886.

Com a punt final remarcar que la mitjana i la variació en cada gràfic es calcula utilitzant totes les imatges disponibles. En altres paraules, ens retornen el valor mitja i la variació d'aquest valor en una determinada posició de la imatge utilitzant els resultats d'aquesta posició en les 448 consultes.

Fins ara s'han vist els resultats globals de l'aplicació, seria interessant per altra banda mostrar alguns exemples d'execució i els resultats obtinguts. La figura 54 mostrarà els resultats d'algunes execucions.



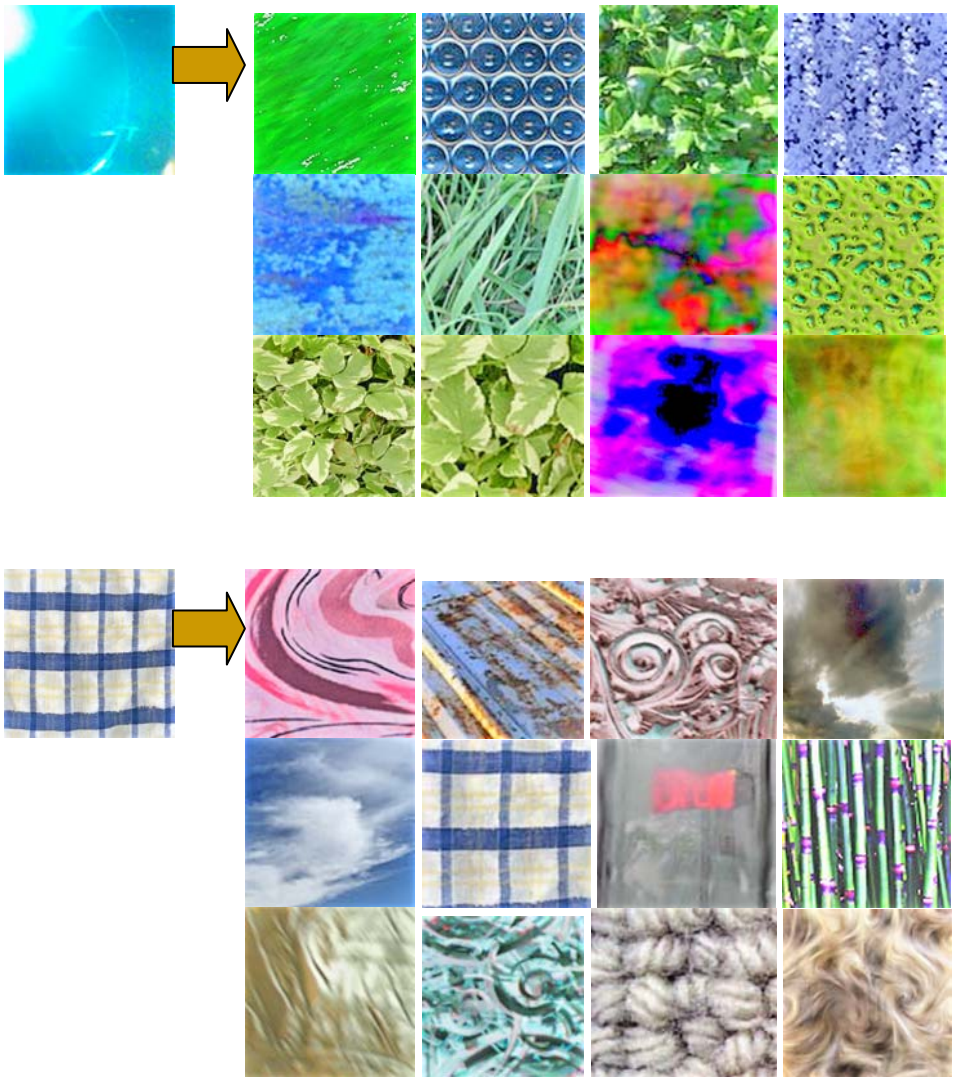


Figura 54: ***Diverses execucions amb els seus resultats de la nostra aplicació.***

Si bé els resultats apareixen en un ordre que aparentment no seria l'esperat cal recordar que els éssers humans primer discriminem per color, l'ordenació dels resultats mostrats és el resultat final de l'aplicació, on l'ordenança que s'efectua és mitjançant la semblança de textura.

6.- Conclusions

Els resultats obtinguts per la nostra implementació són força bons, tot i que no excel·lents. Tenint en compte els objectius que ens plantejàvem "a priori" es podria dir que aquests s'han complert ja que s'ha aconseguit una aplicació que soluciona bastant bé les exigències inicials del problema.

Una solució general òptima al problema és gairebé impossible d'aconseguir, almenys avui dia, ja que pel moment no som capaços d'extreure característiques invariants i clarament diferenciadores de les imatges. Els mètodes amb els quals s'està treballant avui dia ofereixen l'extracció de les característiques més diferenciadores, però aquestes no són suficients com per poder realitzar una cerca perfecta.

L'allau d'algorismes disponibles per a l'extracció de cada una de les característiques diferenciadores (color, forma, textura) i els paràmetres de sintonització que aquests proporcionen, dificulten l'elecció d'aquests. Una possible solució seria realitzar una aplicació mitjançant el paradigma de prova i error per veure quin s'adapta millor al context en el que estem treballant. Les variables de sintonització d'aquests algorismes s'haurien d'ajustar mitjançant un sistema de retro-alimentació amb l'usuari final.

Aquests sistemes requereixen de molts recursos de càlculs, sobretot quan es disposa d'una gran base de dades, aquest ús es pot minimitzar calculant inicialment les característiques d'aquestes i emmagatzemar-les en un fitxer. Com és habitual en informàtica, aquests tipus de sistemes requereixen d'un compromís entre fiabilitat i recursos de còmput. Resultats de l'ordre del 70 % d'encert es poden obtenir amb ordenadors personals, ara bé millors resultats requereixen d'algorismes complexes que s'haurien d'executar en equips molt més potents.

Una curiositat en la que m'he trobat realitzant el projecte, és que molts dels sistemes publicats no poden ser provats en un ordenador personal, no estan disponibles. En la majoria de casos estan darrera un servidor i en cap cas pots modificar les imatges, en el sentit que la cerca es realitza sobre unes imatges preestablertes; aquesta situació planteja dues conclusions, la primera és que el sistema és molt complex i que no pot ser

executat en un ordenador personal o bé el sistema no funciona tan correctament com volen donar a entendre, esta especialitzat per aquestes imatges concretes.

Com s'ha mencionat en repetides ocasions en el projecte, una solució general i òptima encara no existeix, es disposa d'algunes aplicacions, en la majoria de casos comercials, que proporcionen uns resultats espectaculars. Aquestes aplicacions ens porten a la conclusió que és un problema que possiblement té solució i que no estem molt lluny de trobar-la.

Un bon camí per construir una aplicació basada en CBIR i de propòsit general seria dividir les imatges en grups, com per exemple, un grup on sols apareixerien cares, un altre destinat a paisatges i així fins a obtenir diferents grups. Un cop es disposes d'una classificació en grups, el problema deixa de ser tant complex, ja que per par cada grup es poden obtenir un seguit de característiques diferenciadores, i executar per a cada un dels conjunts els algorismes més adients. En altres paraules obtenir la solució òptima final com la descomposició en solucions òptimes de conjunts, on en la majoria de les descomposicions existeix una solució més o menys madura.

Com a conclusió final dir que el sistema humà segueix essent el millor mètode per discriminar segons semblança, ja que no sols utilitza les característiques primàries de la imatge sinó que utilitza també la interpretació i la intel·ligència. Fins que la intel·ligència artificial no solucioni aquests problemes no es podrà parlar d'un veritable sistema CBIR.

7.- Millores

Si bé l'aplicació obté uns resultats raonables, no és si molt menys perfecta, existeixen alguns punts importants que segurament es podrien millorar.

El primer punt seria el temps de còmput de la cerca, fins ara el que fèiem era calcular una a una les característiques de la imatge, el que comportava que l'execució d'una cerca tarda uns 6 minuts en un Intel Dothan 1.75 Mhz amb 1024 MB de memòria DDR-2. No és un temps excessivament gran, però estem parlant sols d'una cerca. Per tal de millorar l'eficiència en aquest aspecte seria interessant canviar la metodologia, un cop s'afegeixen imatges a la base de dades seria interessant extreure'n les característiques i guardar-les en un fitxer. La cerca en aquest cas consistiria en obtenir les característiques d'aquest fitxer i comparar-les amb les de la imatge de petició. Aquest procés implicaria un temps més gran de creació de la base de dades, aproximadament uns 15 segons per imatge, i la cerca trigaria 20 segons aproximadament.

El segon punt interessant seria veure com respon el sistema aplicant altres algorismes, que per motiu de temps no s'han pogut provar, com per exemple el vector de coherència de color. El més possible és que aquests algorismes en millori'n l'eficiència a costa d'un temps més alt de còmput.

El tercer punt i segurament en el que està la clau, és realitzar una etapa prèvia que consistiria en dividir les imatges en grups, de forma que cada un d'aquests grups disposes d'imatges amb característiques més semblants. Posteriorment per a cada grup caldria estudiar quin mètode i sintonització d'aquest dóna millors resultats. Aquesta millora es pot veure com una aplicació pràctica del paradigma "divideix i guanyaràs".

Un quart punt seria acabar de realitzar la comparació amb el 75% restant de la base de dades, i veure d'aquesta forma si els percentatges d'encert es mantenen o milloren o si per contra l'aplicació deixa de ser una solució.

Un cinquè punt seria el tema destinat a l'usuari final, consistiria en afegir una interfície intuïtiva i de fàcil ús per a poder ser executat de forma còmoda. Per altra

banda també seria interessant afegir noves funcionalitats com per exemple permetre que la funció afegir, permetés la inserció d'una nova imatge, o bé la inclusió d'un directori que fos diferent al que esta plantejat per defecte.

Un sisè punt consistiria en especialitzar el programa, consistent en afegir un mecanisme de comunicació entre el programa i l'usuari, de forma que cada execució, segons l'opinió de l'usuari, modifiques alguns dels paràmetres del programa, com per exemple el nombre d'imatges de resultat que retorna en cada pas, o l'ordre en el càlcul dels coeficients de Daubechies.

El punt numero set consisteix en utilitzar els diferents espais de color per a l'extracció de característiques basades en color, el més probable és que els espais HSV i HSB obtinguin uns resultats semblants a la percepció visual humana.

Al ser un problema actual en investigació, és possible que hagin sortit nous mètodes, nous algorismes, i noves conclusions sobre com afrontar-lo. Per aquest motiu seria interessant, en el cas de voler utilitzar aquest projecte com a punt de partida, cercar informació sobre l'estat en que es troba la investigació i adaptar-lo amb la nova informació obtinguda.

8.- Bibliografia

- **[1] Ashley, J., Flickner, M., Hafner, J., Lee, D., Niblack, W., Petkovic, D.** 1995. The query by image content (QBIC) system. *Proceedings of the 1995 ACM SIG MOD International Conference on Management of Data*

- **[2] Li, J., Wang, J.Z., Wiederhold**, G.2000. IRM: Integrated region matching for image retrieval. *Proceedings of the Eight ACM International Conference on Multimèdia* (MULTIMÈDIA'00, Marina del Rey, California, USA)

- **[3] Ma, W. Y., Manjunath, B.S**. 1997. NETRA: A toolbox for navigating large image databases. *Proceedings of the IEEE international Conference on Image Processing* (ICIP, Santa Barbara, California, October)

- **[4] Nasciment, M., Chitkara, V**. 2002. Color-based image retrieval using binary signatures. *Proceedings of the 17th Symposium on Proceedings of the 2002 ACM Symposium on Applied Computing* (SAC'2002, Madrid, Spain)

- **[5] Newsam, S., Sumengen, B.,Manjunath, B.S**. 2001. Category-based image retrieval. *Proceedings of the IEEE International Conference on Image Processing* (ICIP 2001, Thessalonica, Greece)

- **[6] Pass, G., Zabih, R., Miller, J.** 1996. Comparing images using color coherence vectors. *Proceedings of the Fourth ACM International Conference on Multimèdia* (ICM, Boston, Massachusetts)

- **[7] Smuelders, A. W. M., Worring, M., Santini, S., Gupta, A., Jain, R.** 2000. Content-based image retrieval at the end of the early years. *IEEE Transactions on Pattern Analysis and Machine Intelligence 22.* 12 Dec.

- [8] 1995, Automated image retrieval using color and texture. Columbia University technical report, july, 1995

- **[9] Simone santini, Ramesh Jain.** 1999 Similarity Measures. *IEEE transactions on pattern analysis and machine intelligence 21*. September 1999.

- **[10] Eds. C.H. Chen, L.F.Pau, P.S.P. Wang.** Handbook of Pattern Recognition and computer vision.

- **[11] Sharmin Siddique**, 2002. A Wavelet Based Technique for Analysis and Classification of Texture Images. Carleton University, Ottawa, Canada. April 2002.

- **[12] Shengjiu Wang**, 2001, A Robust CBIR Approach Using Local Color Histograms. Department of Computer Science, University of Alberta, Edmonton, Alberta Canada, October 2001.

- **[13] Marinette Bouet, Ali Khenchaf, Henri Briand**. 1999, Shape Representation for image Retrieval. 1999.

- **[14] Kristian Sandberg**. 2000, The Daubechies wavelet transform. Dept. of Applied Mathematics, University of Colorado, Boulder, April 2000

- **[15] Raimondo Schettini, Gianluigi Ciocca, Silvia Zuffi**. 2002. A Survey of Methods for Color Image Indexing and Retrieval in Image Databases. Istituto Tecnologie Infomatiche Multimèdiali, Milano.

- **[16] Atul Sajjanhar, Guojun Lu**. A Grid Based Shape Indexing and Retrieval Method. Gippsland School of Computing and Information Technology, Monash University, Victoria, Australia

- **[17] Tamura, H., Mori, S., Yamawaki, T.** Textural Features Corresponding to Visual Perception.

- **[18] Bakk**, **Medien-Inf. Tilo Burghardt.** Using Sobel Filter Transformation. University of Briston.

- **[19] A.Ben Hanza.** Nonextensive information-theoretic measure for image detection. Concordia Institute for Information System Engineering, Québec, Canada.

RESUM

Les tecnologies actuals han ocasionat una quantitat de recursos multimèdia enorme, obligant a emmagatzemar-los en biblioteques digitals, el que comporta un augment dels recursos disponibles digitalment i en dificulta la seva cerca. La cerca textual ja no és una opció òptima perquè és totalment subjectiva respecte a l'usuari que realitza la cerca. Una solució a aquest problema és realitzar la cerca segons contingut. Aquests sistemes, coneguts com a CBIR, es basen en el paradigma "query by visual example", mostra'm un exemple del resultat que desitges i et retornaré la informació més semblant.

RESUMEN

Las tecnologías actuales han permitido una gran cantidad de recursos multimèdia obligando a que su almacenamiento sea mediante bibliotecas digitales, lo que implica una cantidad inmensa de recursos disponibles, ocasionando que su búsqueda sea muy costosa. La búsqueda textual ya no es una opción óptima porque es totalmente subjetiva. Una posible solución seria realizar la búsqueda mediante el contenido. Dichos sistemas se conocen como CBIR, y se basan en el paradigma "query by visual example", enséñame un ejemplo del resultado y te mostraré los recursos más parecidos.

ABSTRACT

The present technologies have allowed having a huge amount of multimèdia resources, forcing to store them in digital libraries, which implies a high amount of resources available, and an expensive search method. The textual search is no longer an optimal option because this way is totally subjective. A possible serious solution to make the search will be the search using the content. These systems are known as CBIR, and are based on the paradigm "query by visual example", show me a result example and I will show you the optimal similar resources.

Marc Vilana Roca

.....

9.- Manual d'ús

L'execució de l'aplicació consisteix en la crida de dues funcions, la funció afegir i la funció buscar.

La funció afegir és l'encarregada de crear la base de dades amb el nom i la ubicació de totes les imatges que en formen part. Cal dir, però, que per motius de temps sols s'ha realitzat la part per a afegir un únic directori, el directori per defecte. Per a que aquesta funció realitzi correctament la tasca, les imatges han d'estar contingudes dins el directori \imatge, en la mateixa carpeta on es troben els fitxers executables. L'execució d'aquesta funció seria *afegir('',0);* .Els resultats o la numeració de les imatges que formen part es generaran en l'arxiu dadesimatges.txt.

Cal dir que la funció afegir no realitza l'etapa de normalització de les imatges, aquest procés s'ha de realitzar amb anterioritat a la crida. Dins l'annex es troba el codi necessari per a realitzar aquesta etapa de normalització.

És important respectar la nomenclatura del nom de la imatge, en altres paraules el nom de la imatge ha de ser estrictament 5 lletres, aquest detall és important pel punt següent.

El segon pas consisteix en la crida a la funció buscar. Aquesta funció rep com a paràmetres la ubicació exacta de la imatge que utilitzarem com a referència, en altres paraules, la imatge de la qual volem obtenir les més semblants. La crida és realitza afegint la direcció absoluta on es troba aquesta, com per exemple: *buscar ('c:\Documents and Settings\crash\escritorio\project\imatge\im379.bmp');* . Aquesta funció generarà tres arxius. Im379color.txt, im379contorn.txt i im379textura.txt, les 5 primeres lletres representen el nom de la imatge per a generar l'arxiu, per aquest motiu s'exigia la diferenciació d'imatges amb exactament 5 lletres. Pel que fa al contingut de l'arxiu els fitxers color fan referència als resultats obtinguts en l'etapa d'extracció de característiques mitjançant el color; els arxius contorn ens mostren informació sobre el resultat d'aplicar l'extracció de característiques de contorn als resultats obtinguts per l'etapa de color; i finalment els arxius textura són el resultat de combinar les tres etapes, en altres paraules són els resultats finals de l'aplicació.

10.- Annex

10.1.- Codi per a realitzar les normalitzacions

```
directori = pwd;
imatges = '\imatge\';
for (i=1:length(imatges))
   directori(length(pwd)+i) = imatges(i);
end

 for (i=1:4)
   switch (i)
      case 1
            extencio = '*.bmp';
      case 2
            extencio = '*.jpg';
      case 3
            extencio = '*.pgm';
      case 4
            extencio = '*.pcx';
      otherwise
            disp('error de format')
            return;
   end
   llistaArxius = dir([directori extencio]);
   [X,Y] = size(llistaArxius);

   for numIma = 1:X
      llistaArxius(numIma).name
      imres = indane(imread([directori llistaArxius(numIma).name]));
      imres = imresize(imres,[256,256],'bicubic');
      imwrite(imres,([directori llistaArxius(numIma).name]),extencio(3:5));
   end
 end
```

10.2.- Codi per a llegir les distancies del tots contra tots

```
function [color,contorn,textura ] = lectura(i)
fi1 = 'color.txr';
fi2 = 'contorn.txr';
fi3 = 'textura.txr';
  if (i <100)
    inici = 'im0';
  else
    inici = 'im';
  end
col = sprintf('%s%i%s' ,inici,i, fi1);
con = sprintf('%s%i%s' ,inici,i, fi2);
txtu = sprintf('%s%i%s' ,inici,i, fi3);
fid = fopen(col);
color = fscanf(fid, '%f)';
fclose(fid);
fid = fopen(con);
contorn = fscanf(fid, '%f)';
fclose(fid);
fid = fopen(txtu);
textura = fscanf(fid, '%f)';
fclose(fid);
```

10.3.- Codi per guardar la distància de tots contra tots

```
fid = fopen('resultats.m','w+');

for i=1:448
    crida =sprintf('%s%i%s','[a,b,c] = lectura(',i,');');
    fprintf(fid,'%s \r',crida);
    arxiu = sprintf('%s%i%s','color(',i,',:) = a;' );
    fprintf(fid,'%s \r',arxiu);
    arxiu = sprintf('%s%i%s','contorn(',i,',:) = b;' );
    fprintf(fid,'%s \r',arxiu);
    arxiu = sprintf('%s%i%s','textura(',i,',:) = c;' );
    fprintf(fid,'%s \r',arxiu);
    fprintf(fid,'\r');
end

fprintf(fid, '%s','save vector.mat');

fclose(fid)
```

10.4.- Codi per a realitzar les proves tots contra tots

```
% dadesimatges2.txt = dadesimatges.txt
fod = fopen('dadesimatges2.txt');

while 1
    imatge = fgetl(fod);
    imatge
    if ~ischar(imatge), break, end       % Meaning: End of File..


nom = imatge(length(imatge)-8: length(imatge)-4);
nom1 = nom;

txt_color = 'color.txt';
    for (i=1:length(txt_color))
            nom(length(nom1)+i) = txt_color(i);
    end
txt_color = nom;
txt_colorr = nom;
txt_colorr(length(txt_colorr+1)) = 'r'

nom = nom1;

txt_contorn = 'contorn.txt';
    for (i=1:length(txt_contorn))
            nom(length(nom1)+i) = txt_contorn(i);
    end

txt_contorn = nom;
txt_contornr = nom;
txt_contornr(length(txt_contornr+1)) = 'r'

nom = nom1;

txt_textura = 'textura.txt';
    for (i=1:length(txt_textura))
            nom(length(nom1)+i) = txt_textura(i);
    end
```

```
txt_textura = nom;
txt_texturar = nom;
txt_texturar(length(txt_texturar+1)) = 'r'

nom = nom1;

total = 0;
fid = fopen('dadesimatges.txt');
resultValues = [];      % Results matrix...
resultNames = {};
i = 1;                  % Indices...
j = 1;
[img,HSVmapq] = imread(imatge);
[k1,m1] = kmeans(img,4);
Xq = img;


while 1
    imagename = fgetl(fid);
    if ~ischar(imagename), break, end       % End of File...
    total = total +1;
    %if ( strcmp(imagename,imatge) ~= 1  ) % comparar amb ella ¿?
        [img2, HSVmap] = imread(imagename);
        [k2,m2] = kmeans(img2,4);
        imagename
        X = img2;
        D3= mahalanobis(k1,k2); %clusterin

        %D3 = cof_fourier(Xq,X);

        resultValues(i) = abs(D3);
        resultNames(j) = {imagename};

        i = i + 1;
        j = j + 1;
    % end
end

fclose(fid);

% Sorting colour results...

[sortedValues, index] = sort(resultValues);

fid = fopen(txt_color, 'w+');
fot = fopen(txt_colorr,'w+');

for i = 1:ceil(total/8)
    tempstr = char(resultNames(index(i)));
    fprintf(fot, '%f ', resultValues(index(i)) );
    fprintf(fid, '%s\r', tempstr);
end
fclose(fid);
fclose(fot);

% contourn search...
disp('contorn')
fid = fopen(txt_color);
resultValues = [];      % Results matrix...
resultNames = {};
i = 1;                  % Indices...
```

```
j = 1;

Xq = img;

while 1
    imagename = fgetl(fid);
    if ~ischar(imagename), break, end       % End of File...

        [img2, HSVmap] = imread(imagename);
        imagename
        X = img2;

       %D3= mahalanobis(k1,k2); %clustering
       %D4 = calc_grad(Xq, X); % gradient
        D5 = semblant(Xq,X);

        resultValues(i) = abs(D5);
        resultNames(j) = {imagename};
        i = i + 1;
        j = j + 1;
end

fclose(fid);


% Sorting contorn results...

[sortedValues, index] = sort(resultValues);

fid = fopen(txt_contorn, 'w+');
fot = fopen(txt_contornr, 'w+');

for i = 1:ceil(total/16)        % Store top 27 matches...
    tempstr = char(resultNames(index(i)));
    fprintf(fot, '%f ', resultValues(index(i)) );
    fprintf(fid, '%s\r', tempstr);
end
fclose(fid);
fclose(fot);


% textura
disp('texture')

 [Xq,HSVmapq] = imread(imatge);
 queryEnergies = obtainEnergies(Xq, 6);

 fid = fopen(txt_contorn);

fresultValues = [];      % Results matrix...
fresultNames = {};
i = 1;                  % Indices...
j = 1;

while 1
    imagename = fgetl(fid);
    if ~ischar(imagename), break, end       % End of File...

    [X, RGBmap] = imread(imagename);
    imagename
    imageEnergies = obtainEnergies(X, 6);
```

```
     E = mahalanobis(queryEnergies, imageEnergies);

    fresultValues(i) = E;
    fresultNames(j) = {imagename};
    i = i + 1;
    j = j + 1;
end

% Sorting final results...

[sortedValues, index] = sort(fresultValues);
fid = fopen(txt_textura, 'w+');
fot = fopen(txt_texturar, 'w+');


for i = 1:ceil(total/32)
    imagename = char(fresultNames(index(i)));
    fprintf(fid, '%s %f\r', imagename, fresultValues(index(i)) );
    fprintf(fot, '%f ', fresultValues(index(i)) );
end

fclose(fot);
fclose(fid);

end
fclose(fod);
```

10.5.- Codi per a realitzar gràfiques de distàncies

```
clear all
close all
load vector.mat
resultat = [];
mitjana = [];
for i=1:56
   resultat(i) = std2(color(:,i)');
   mitjana(i) = mean2(color(:,i)');
end
figure(),plot(resultat),title('desviacio c')
figure(),plot(mitjana),title('mitjana c')
resultat = [];
mitjana = [];

for i=1:28
   resultat(i) = std2(contorn(:,i)');
   mitjana(i) = mean2(contorn(:,i)');
end
figure(),plot(resultat),title('desviacio con')
figure(),plot(mitjana),title('mitjana con')
resultat = [];
mitjana = [];
for i=1:14
   resultat(i) = std2(textura(:,i)');
   mitjana(i) = mean2(textura(:,i)');
end
figure(),plot(resultat),title(' desv textura')
figure(),plot(mitjana),title('mitjana textura')
```