

**Oil Spill’s Features Extraction**

**Progetto di Image Processing**

**Vito Domenico Tagliente**

**Pietro Tota**

**Luca Riccardi**

**1. Introduzione**

Nel presente elaborato vengono proposte le metodologie e le tecniche che sono state adottate per la realizzazione di un sistema automatico in grado di svolgere il compito di “Features extraction” (estrazione delle caratteristiche) nel campo dell’oil spill detection.

Questa fase del processo elaborativo delle immagini è molto importante in quanto permette con facilità, fornito un insieme di immagini classificate(oil o lookalike), di estrarre diverse informazioni che verranno adoperate per la creazione di un trainig set sul quale verrà definito un modello. Tale modello verrà in seguito adoperato per la classificazione di successivi dataset di immagini al fine di, definita la conoscenza di base del modello, distinguere le immagini contenti macchie di petrolio dai falsi positivi, ovvero immagini contenti macchie “lookalike”, simili.

Nella fase di creazione del training set non si è tenuto conto delle caratteristiche ancellari: presenza di navi o caratteristiche del vento durante il momento dell’acquisizione dell’immagine. Tali caratteristiche possono essere inserite in un approfondimento futuro(lavori futuri?)

**2. Segmentazione**

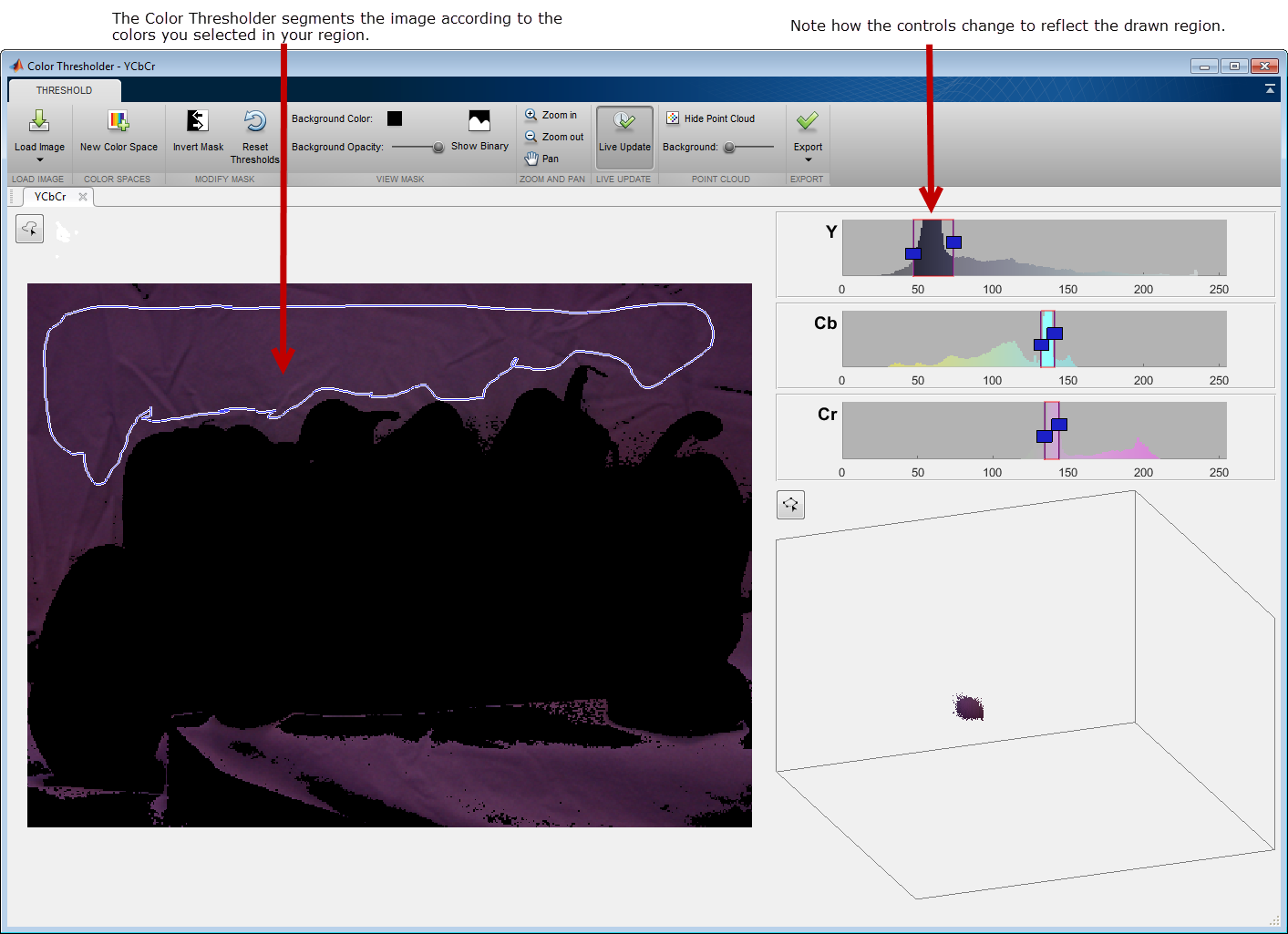
La fase di Segmentazione è stata tralasciata dal contesto di studio per la realizzazione di questo progetto, pertanto è stato possibile semplificare l’operazione concentrando la forza lavoro del team per lo studio e la ricerca delle features.

Come già noto, le tecniche di segmentazione possono essere di due tipologie:

* Manuali, basate sull’esperienza dell’operatore
* Automatizzate

https://it.mathworks.com/help/images/colorapp_app_button.pngIn questo contesto è stato pensato di definire una segmentazione manuale basata sull’utilizzo degli strumenti di elaborazione di immagini che Matlab fornisce, il Color Thresholder.

Tramite l’utilizzo di questo toolbox, è stato possibile, adoperando delle operazioni manuali di selezione delle aree dello sfondo di segmentare l’immagine, nel nostro caso la macchia di petrolio.



L’efficienza del tool proposto consiste nella possibilità di convertire la sequenza di operazioni svolte in modalità manuale in uno script, riutilizzabile per studi successivi sulla stessa immagine.

**3. Features extraction**

Per la realizzazione del progetto in esame, si è pensato di suddividere la complessità del problema in modo tale da isolare il codice per categoria (features Geometriche, Backscatter, Texture) e soprattutto per rendere la struttura del progetto modulare e garantirne la manutenibilità.

C:\Users\HackJ\AppData\Local\Microsoft\Windows\INetCache\Content.Word\scriptsuml.png

Il grafico sopra mostrato rappresenta le relazioni che intercorrono tra i vari script definiti in Matlab, in particolare si evidenziano visivamente le tre macro categorie di features sopra citate:

* Geometriche
* Backscatter
* Texture

**4. Preparazione del Workspace**

Esaminiamo, ora, le istruzioni preliminari che l’algoritmo realizzato richiede prima di svolgere le proprie elaborazioni. Tale fase, quindi, si occupa di caricare nel workspace di Matlab le variabili che verranno successivamente processate.

Tali istruzioni sono facilmente richiamabili utilizzando lo script “*>> spillbegin*”, nel dettaglio otteniamo il seguente listato:

% Utilizza questo script per preparare il Workspace di Matlab

% img: immagine originale

img **=** imread**(** 'oil.bmp' **);**

% gimg: immagine non segmentata in scala di grigi

gimg **=** rgb2gray**(** img **);**

% s: immagine segmentata

s **=** spillseg**(** img **);**

% back: immagine di background

back **=** spillback**(** img**,** s **);**

% gback: immagine di background in scala di grigi

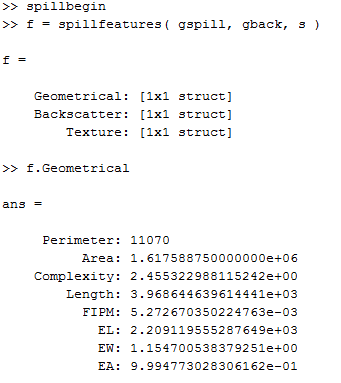
gback **=** rgb2gray**(** back **);**

% gspill: macchia di petrolio in scala di grigi

gspill **=** gimg **-** gback**;**

**5. Estrazione delle caratteristiche**

La fase di estrazione delle features, per come è stato modellato e strutturato il progetto, prevede l’utilizzo di un solo comando “f = spillfeatures( gspill, gback, s)”. Tale funzione, dopo aver elaborato i dati di input, ritorna una struttura dati in cui sono presenti tutte le features calcolate.



Come mostrato, tutta l’elaborazione avviene in maniera strutturata e modulare.

**6. Geometrical features**

In questa sezione del progetto ci siamo occupati di estrarre tutte le caratteristiche di natura geometrica. Esaminiamo dapprima il contenuto dello script “spillgeom”.

% Script per il calcolo delle features di natura geometrica

% img: Spill oil segmentato

**function** **[** out **]** **=** spillgeom**(** img **)**

% 1. Calcolo del perimetro

**[**out**.**Perimeter**,** perim\_img**]** **=** spillperim**(** img **);**

% 2. Calcolo dell'area

out**.**Area **=** spillarea**(** img **);**

% 3. Calcolo della complessità Dell'oggetto

% This feature will take a small numerical value for regions with simple

% geometry and larger values for complex geometrical regions.

out**.**Complexity **=** out**.**Perimeter **/** **(**2 **\*** sqrt**(** pi **\*** out**.**Area **));**

% 4. Length (L): sum of skeleton edges

% (obtained by Delaunay triangulation),

% that build the main line.

out**.**Length **=** spilldelaunay**(** img **);**

% Width (W): mean value of Delaunay triangles

% which are crossed by main line.

% out.Width = ?

% Length To Width Ratio (LWR)

% out.LWR = out.Length / out.Width;

% Compactness (Comp), defined as

% out.Comp = ( out.Length \* out.Width ) / out.Area;

% 5. Calcolo del FIPM

out**.**FIPM **=** spillfipm**(** img **);**

% 6. Calcolo dei parametri basati sull fitting dell'ellisse

% Ellipse Length: value of main axe of an ellipse fitted to the data

% Ellipse Width: value of minor axe of an ellipse fitted to the data.

**[** out**.**EL**,** out**.**EW **]** **=** spillellipfit**(** img **);**

% Ellipse Asymetry

out**.**EA **=** 1 **-** **(** out**.**EW **/** out**.**EL **);**

**6.1 Calcolo del perimetro**

% Questo script si occupa di ricavare il perimetro

% dell'oggetto in input

**function** **[**length**,** perim\_img**]** **=** spillperim**(** img**,** debug **)**

% il parametro debug è opzionale

**if** nargin **<=** 1

debug **=** false**;**

**end**

% L'oggetto in input deve essere binario

temp **=** spillbin**(** img **);**

% Produco l'immagine contenete il bordo dell'oggetto

perim\_img **=** bwperim**(** temp **);**

% Debug grafico

**if** debug

imshow**(**perim\_img**);**

**end**

% Calcolo il perimetro

length **=** sum**(** int32**(** perim\_img**(:)** **)** **);**

Notiamo che in questo script è stata utilizzata la funzione spillbin, tale funzione si occupa di verificare che la matrice in input sia binaria, in caso negativo provvede con la binarizzazione.

imgbin **=** img**;**

**if** size**(**img**,** 3**)** **==** 3 % vuol dire che l'immagine è RGB

level **=** graythresh**(**img**);**

imgbin **=** im2bw**(**img**,** level**);**

**end**

**6.2 Calcolo dell’area**

Riguardo a questo calcolo, l’utilizzo di Matlab come strumento di sviluppo ci ha agevolati molto, in quanto esistono moltissime funzioni predefinite per la soluzione di problemi banali.

% Calcolo dell'area dell'oggetto passato in input

**function** **[**area**]** **=** spillarea**(** img **)**

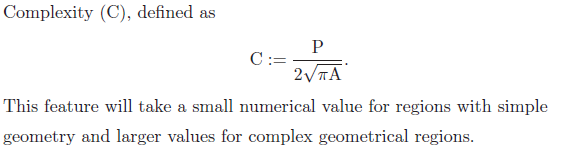
% L'oggetto in input deve essere binario

temp **=** spillbin**(** img **);**

% Calcolo dell'area

area **=** bwarea**(** temp **);**

**6.3 Complessità Geometrica**



Tale caratteristica viene calcolata internamente allo script *spillgeom*

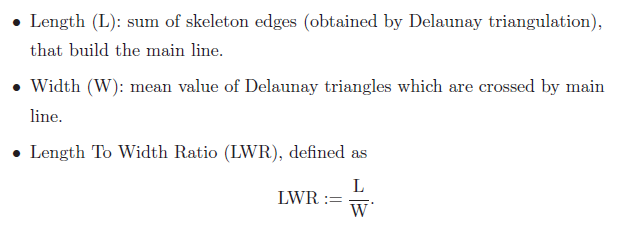
% This feature will take a small numerical value for regions with simple

% geometry and larger values for complex geometrical regions.

out**.**Complexity **=** out**.**Perimeter **/** **(**2 **\*** sqrt**(** pi **\*** out**.**Area **));**

**6.4 Triangolazione di Delaunay**

Questo script calcola diversi parametri geometrici utilizzando la triangolazione di Delaunay.



Contando che nel presente lavoro non siamo riusciti ad estrarre la caratteristica “width” e di conseguenza LWR, che per il momento restano in sospeso.

Al contrario siamo riusciti ad estrarre il parametro Length in questo modo:

%Script che calcola la lenght usando la triangolazione di Delaunay

**function** **[**d**]** **=** spilldelaunay**(** img**,** debug **)**

% il parametro debug è pzionale

**if** nargin **<=** 1

debug **=** false**;**

**end**

%Inizio dello script. Viene passato in input l'immagine s

**[**x**,**y**]=** getvectors**(**img**);**

x**=**x**';** % la funzione delaunayTriangolation vuole vettori colonna

y**=**y**';**

DT **=** delaunayTriangulation**(**x**,**y**);**

**if** debug

triplot**(**DT**);**

axis equal

xlabel**(**'Longitude'**),** ylabel**(**'Latitude'**)**

grid on

hold on

**end**

**[**a**,**b**]=** size**(**DT**.**Points**);**%

DT**.**Points**(**a**,:)=[];**%elimina l'ultimo elemento della matrice DT.Points che è il punto(0,0) che aggiunge la funzione di default(non è previsto)

k **=** convexHull**(**DT**);**%Contiene l'indice di riga degli elementi del vettore DT.Points che definiscono il contorno.

xHull **=** DT**.**Points**(**k**,**1**);**%Crea il vettore di ascisse con i punti del vettore DT.Points

yHull **=** DT**.**Points**(**k**,**2**);**

**if** debug

plot**(**xHull**,**yHull**,**'r'**,**'LineWidth'**,**2**);**

hold off

**end**

d**=**0**;**

**for** i**=**1 **:** size**(**xHull**)-**1 % ciclo che calcola la distanza euclidea tra due punti usando la distanza euclidea

d**=** d **+** sqrt**((**xHull**(**i**)-**xHull**(**i**+**1**))^**2 **+** **(**yHull**(**i**)-**yHull**(**i**+**1**))^**2**);**

**end**

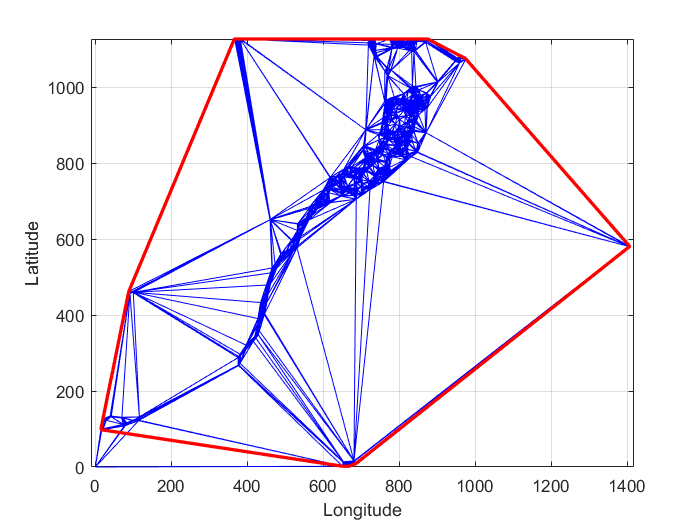
% %ti = edgeAttachments(DT,k(1),k(2));

% %ti{:};%righe della connectivity List contenente il record dei vertici del triangolo

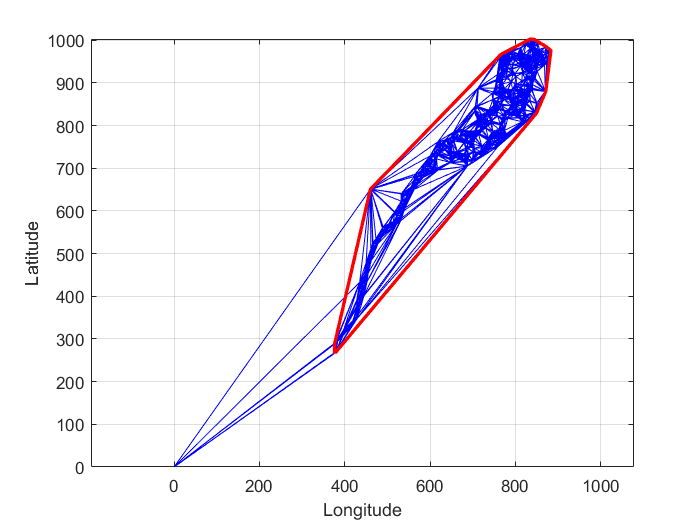
% val= DT.ConnectivityList(ti);

% val{:};

Nelle immagini seguenti è rappresentato il plot dei calcoli eseguiti dallo script, ottenibile impostando il parametro debug a TRUE.



Da notare che questa immagine è stata segmentata col tool di matlab, quindi presenta degli elementi spuri nei dintorni della macchia di petrolio. Segmentandola manualmente, eseguendo delle operazioni di pulizia mirata, si ottiene il seguente plot.

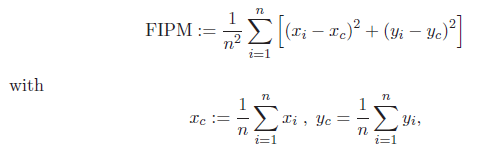


Il contorno rosso rappresenta il parametro length.

Da notare che entrambe le immagini risultano ruotate di 180° rispetto all’immagine originale. Eliminando il punto (0,0), che la funzione della triangolazione di delaunay aggiunge di default, notiamo che il contorno rosso individuato, in questo caso, abbraccia meglio la macchia di petrolio.

**6.5 First Invariant Planar Moment (FIPM)**

Tale feature è definita matematicamente dalla seguente formalismo:



La conoscenza della formula completa ci ha agevolato nella scrittura di questo script:

% First Invariant Planar Moment (FIPM)

**function** **[**FIPM**]** **=** spillfipm**(** img **)**

% n the number of points in the dark patch contour.

% Quindi lavoriamo solo con i pixel del contorno

**[**length**,** perim**]** **=** spillperim**(** img **);**

% Numero di pixel pari a 1 dell'immagine

n **=** numel**(** perim**(:)** **);**

% Dimensioni dell'immagine

**[**nrows**,** ncols**]** **=** size**(** img **);**

% Calcolo xc e yc

xc **=** 0**;**

yc **=** 0**;**

**for** j **=** 1**:**ncols

**for** i **=** 1**:**nrows

**if** perim**(**i**,** j**)** **==** 1

xc **=** xc **+** i**;**

yc **=** yc **+** j**;**

**end**

**end**

**end**

xc **=** xc **/** n**;**

yc **=** yc **/** n**;**

% Calcolo di FIPM

FIPM **=** 0**;**

**for** j **=** 1**:**ncols

**for** i **=** 1**:**nrows

**if** perim**(**i**,** j**)** **==** 1

FIPM **=** FIPM **+** **(** **(**i **-** xc**)^**2 **+** **(**j **-** yc**)^**2 **);**

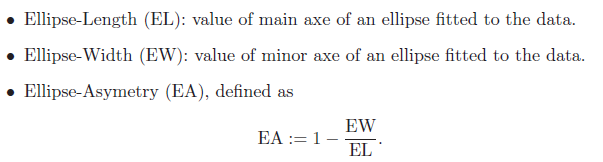
**end**

**end**

**end**

FIPM **=** FIPM **/** **(** n **^** 2 **);**

**6.6 Ellipse fitting**



In questo caso ci siamo occupati di disegnare delle ellissi all’interno della nostra macchia di petrolio, interessandoci dell’asse maggiore e minore tra tutti quelli ritrovati.

% Script per il fitting di una ellisse

% length: value of main axe of an ellipse fitted to the data.

% width: value of minor axe of an ellipse fitted to the data.

**function** **[**majoraxis**,** minaxis**]** **=** spillellipfit**(** img**,** debug **)**

% il parametro debug 蠯pzionale

**if** nargin **<=** 1

debug **=** false**;**

**end**

**[**p**,** pimg**]** **=** spillperim**(** img **);**

props **=** regionprops**(**pimg**,** 'Orientation'**,** 'MajorAxisLength'**,** ...

'MinorAxisLength'**,** 'Eccentricity'**,** 'Centroid'**);**

% Rappresentazioni grafiche delle ellissi

**if** debug

imshow**(** pimg **);**

hold on**;**

phi **=** linspace**(**0**,**2**\***pi**,**50**);**

cosphi **=** cos**(**phi**);**

sinphi **=** sin**(**phi**);**

**for** k **=** 1**:**length**(**props**)**

xbar **=** props**(**k**).**Centroid**(**1**);**

ybar **=** props**(**k**).**Centroid**(**2**);**

a **=** props**(**k**).**MajorAxisLength**/**2**;**

b **=** props**(**k**).**MinorAxisLength**/**2**;**

theta **=** pi**\***props**(**k**).**Orientation**/**180**;**

R **=** **[** cos**(**theta**)** sin**(**theta**)**

**-**sin**(**theta**)** cos**(**theta**)];**

xy **=** **[**a**\***cosphi**;** b**\***sinphi**];**

xy **=** R**\***xy**;**

x **=** xy**(**1**,:)** **+** xbar**;**

y **=** xy**(**2**,:)** **+** ybar**;**

plot(x,y,'r','LineWidth',2);

end

hold off

end

majoraxis = props(1).MajorAxisLength;

minaxis = props(1).MinorAxisLength;

for k = 2:length(props)

if props(k).MajorAxisLength > majoraxis

majoraxis = props(k).MajorAxisLength;

end

if props(k).MinorAxisLength < minaxis

minaxis = props(k).MinorAxisLength;

end

end

Anche in questo caso, abbiamo reso disponibile la modalità di debug al fine di permettere la visualizzazione su grafico delle elaborazioni eseguite.



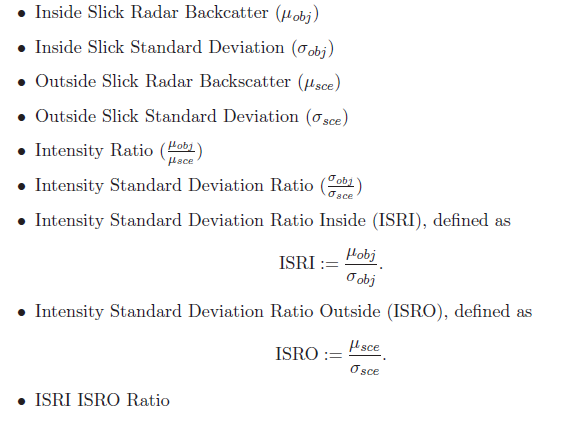
Pertanto il terzo parametro si è ricavato banalmente in *spillgeom* come:

% Ellipse Asymetry

out**.**EA **=** 1 **-** **(** out**.**EW **/** out**.**EL **);**

**7. Backscatter features**

Sono state estratte le seguenti caratteristiche:



**function** **[** out **]** **=** spillbackscatter**(** spill**,** back **)**

% Inside Slick Standard Deviation

out**.**SpillStandardDeviation **=** mean **(**std**(** double**(** spill **)** **)** **);**

% Inside Slick Radar Backcatter

out**.**SpillMean **=** mean**(** mean**(** spill **)** **);**

% Outside Slick Standard Deviation (?sce)

out**.**BackStandardDeviation **=** mean**(** std**(** double**(** back **)** **)** **);**

% Outside Slick Radar Backscatter (?sce)

out**.**BackMean **=** mean**(** mean**(** back **)** **);**

% Intensity Ratio

out**.**IntensityRatio **=** out**.**SpillMean **/** out**.**BackMean**;**

% Intensity Standard Deviation Ratio

out**.**ISDR **=** out**.**SpillStandardDeviation **/** out**.**BackStandardDeviation**;**

% Intensity Standard Deviation Ratio Inside (ISRI)

out**.**ISRI **=** out**.**SpillMean **/** out**.**SpillStandardDeviation**;**

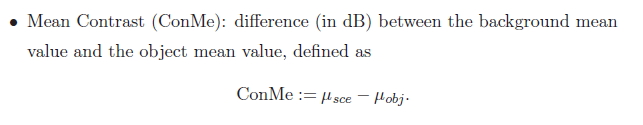
% Intensity Standard Deviation Ratio Outside (ISRO)

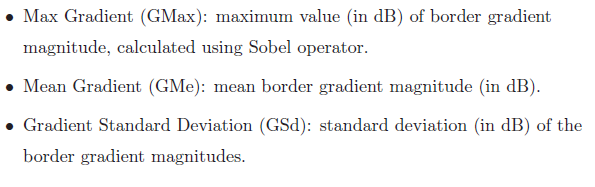
out**.**ISRO **=** out**.**BackMean **/** out**.**BackStandardDeviation**;**

% ISRI ISRO Ratio

out**.**IRatio **=** out**.**ISRI **/** out**.**ISRO**;**

Continuando sono state estratte anche le restanti:





% Mean Contrast (ConMe)

% difference (in dB) between the background mean

% value and the object mean value

out**.**ConMe **=** out**.**BackMean **-** out**.**SpillMean**;**

%Max Gradient (GMax): maximum value (in dB)

% of border gradient magnitude,

% calculated using Sobel operator.

% Mean Gradient (GMe): mean border gradient

% magnitude (in dB).

% Gradient Standard Deviation (GSd):

% standard deviation (in dB) of the

% border gradient magnitudes.

**[**out**.**GMax**,** out**.**GMe**,** out**.**GSd**]** **=** spillgradient**(** spill **);**

**7.1 spillgradient**

Questo script presenta una funzione che prende in input una immagine binarizzata o grayscale e ne calcola il gradiente lungo gli assi x e y (Gx e Gy). Successivamente calcola il G magnitude e la direzione, individuandone il Massimo, la media e la deviazione standard restituendo i valori in dB.

**function** **[** gMax**,** gMe**,** gSt**]** **=** spillgradient**(**I**)**

%Calcolo il gradiente lungo le due direzioni

%[Gx, Gy] = imgradientxy(I);

%calcolo il G Magnitude e la direzione usando sobel (default, Gx e Gy,

%vengono calcolati dentro imgradient se non presenti...

**[**Gmag**,** Gdir**]** **=** imgradient**(**I**,** 'sobel'**);**

%individuo il valore massimo

**[**Max**,**k**]=**max**(**max**(**Gmag**));**

% effettua la media dei valori del perimetro(esclude i valori che sono 0

% nel background dell'immagine! I cicli non tengono conto del

**[**M**,**N**]=** size**(**I**);** %M n. di righe, N n. di colonne

z**=**0**;**

x**=**0**;**

**for** i **=** 1**:**M %riga

**for** j **=** 1**:**N %colonna

**if** **(**Gmag**(**i**,**j**)** **~=** 0**)**

x**=**x**+** Gmag**(**i**,**j**);**

z**=**z**+**1**;**

**end**

**end**

**end**

media**=**x**/**z**;**

%deviazione standard

x**=**0**;**

**for** i **=** 1**:**M %riga

**for** j **=** 1**:**N %colonna

**if** **(**Gmag**(**i**,**j**)** **~=** 0**)**

x**=**x**+(**Gmag**(**i**,**j**)-** media**)^**2**;**

**end**

**end**

**end**

dev**=** sqrt**(**x**/(**z**-**1**));**

%trasformazione in db dei valori trovati

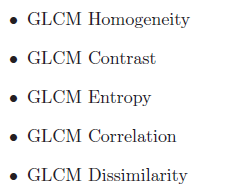
gMax **=** mag2db**(**Max**);**

gMe**=** mag2db**(**media**);**

gSt**=** mag2db**(**dev**);**

**8. Texture features**

In questo caso abbiamo utilizzato delle matrici di co-occorrenza in livelli di grigio (GLCM) per la estrazione di caratteristiche basate sulle texture:



A tal proposito la documentazione di Matlab è risultata più che soddisfacente, in quanto presenta diverse funzioni pre-impostate per la definizione di tali elaborazioni.

Il tutto viene eseguito tramite lo script spilltexture, che si occupa di estrarre tutte le caratteristiche sopra definite:

% Calcolo dei parametri basati sulla Matrice di

% co-occorrenza dei livelli di grigio

**function** **[** out **]** **=** spilltexture**(**img**)**

out**.**GLCM **=** graycomatrix**(**img**);**

% Codice base di Matlab

% stats = graycoprops(glcm);

% stats.Homogeneity

% Le features sono state calcolate utilizzando

% uno script esterno

features **=** glcmfeatures**(** out**.**GLCM**,** 0 **);**

% 1. Homogeneity

out**.**Homogeneity **=** features**.**homom**;**

% 2. Contrast

out**.**Contrast **=** features**.**contr**;**

% 3. Entropy

out**.**Entropy **=** features**.**entro**;**

% 4. Correlation

out**.**Correlation **=** features**.**corrm**;**

% 5. Dissimilarity

out**.**Dissimilarity **=** features**.**dissi**;**

**8.1 glcmfeatures**

Per l’estrazione delle features glcm è stato utilizzato uno script ad alta affidabilità ritrovato sulla sezione di scambio di materiale del sito ufficiale di Matlab.

**function** **[**out**]** **=** glcmfeatures**(**glcmin**,** pairs**)**

%

% GLCM\_Features1 helps to calculate the features from the different GLCMs

% that are input to the function. The GLCMs are stored in a i x j x n

% matrix, where n is the number of GLCMs calculated usually due to the

% different orientation and displacements used in the algorithm. Usually

% the values i and j are equal to 'NumLevels' parameter of the GLCM

% computing function graycomatrix(). Note that matlab quantization values

% belong to the set {1,..., NumLevels} and not from {0,...,(NumLevels-1)}

% as provided in some references

% http://www.mathworks.com/access/helpdesk/help/toolbox/images/graycomatrix

% .html

%

% Although there is a function graycoprops() in Matlab Image Processing

% Toolbox that computes four parameters Contrast, Correlation, Energy,

% and Homogeneity. The paper by Haralick suggests a few more parameters

% that are also computed here. The code is not fully vectorized and hence

% is not an efficient implementation but it is easy to add new features

% based on the GLCM using this code. Takes care of 3 dimensional glcms

% (multiple glcms in a single 3D array)

%

% If you find that the values obtained are different from what you expect

% or if you think there is a different formula that needs to be used

% from the ones used in this code please let me know.

% A few questions which I have are listed in the link

% http://www.mathworks.com/matlabcentral/newsreader/view\_thread/239608

%

% I plan to submit a vectorized version of the code later and provide

% updates based on replies to the above link and this initial code.

%

% Features computed

% Autocorrelation: [2] (out.autoc)

% Contrast: matlab/[1,2] (out.contr)

% Correlation: matlab (out.corrm)

% Correlation: [1,2] (out.corrp)

% Cluster Prominence: [2] (out.cprom)

% Cluster Shade: [2] (out.cshad)

% Dissimilarity: [2] (out.dissi)

% Energy: matlab / [1,2] (out.energ)

% Entropy: [2] (out.entro)

% Homogeneity: matlab (out.homom)

% Homogeneity: [2] (out.homop)

% Maximum probability: [2] (out.maxpr)

% Sum of sqaures: Variance [1] (out.sosvh)

% Sum average [1] (out.savgh)

% Sum variance [1] (out.svarh)

% Sum entropy [1] (out.senth)

% Difference variance [1] (out.dvarh)

% Difference entropy [1] (out.denth)

% Information measure of correlation1 [1] (out.inf1h)

% Informaiton measure of correlation2 [1] (out.inf2h)

% Inverse difference (INV) is homom [3] (out.homom)

% Inverse difference normalized (INN) [3] (out.indnc)

% Inverse difference moment normalized [3] (out.idmnc)

%

% The maximal correlation coefficient was not calculated due to

% computational instability

% http://murphylab.web.cmu.edu/publications/boland/boland\_node26.html

%

% Formulae from MATLAB site (some look different from

% the paper by Haralick but are equivalent and give same results)

% Example formulae:

% Contrast = sum\_i(sum\_j( (i-j)^2 \* p(i,j) ) ) (same in matlab/paper)

% Correlation = sum\_i( sum\_j( (i - u\_i)(j - u\_j)p(i,j)/(s\_i.s\_j) ) ) (m)

% Correlation = sum\_i( sum\_j( ((ij)p(i,j) - u\_x.u\_y) / (s\_x.s\_y) ) ) (p[2])

% Energy = sum\_i( sum\_j( p(i,j)^2 ) ) (same in matlab/paper)

% Homogeneity = sum\_i( sum\_j( p(i,j) / (1 + |i-j|) ) ) (as in matlab)

% Homogeneity = sum\_i( sum\_j( p(i,j) / (1 + (i-j)^2) ) ) (as in paper)

%

% Where:

% u\_i = u\_x = sum\_i( sum\_j( i.p(i,j) ) ) (in paper [2])

% u\_j = u\_y = sum\_i( sum\_j( j.p(i,j) ) ) (in paper [2])

% s\_i = s\_x = sum\_i( sum\_j( (i - u\_x)^2.p(i,j) ) ) (in paper [2])

% s\_j = s\_y = sum\_i( sum\_j( (j - u\_y)^2.p(i,j) ) ) (in paper [2])

%

%

% Normalize the glcm:

% Compute the sum of all the values in each glcm in the array and divide

% each element by it sum

%

% Haralick uses 'Symmetric' = true in computing the glcm

% There is no Symmetric flag in the Matlab version I use hence

% I add the diagonally opposite pairs to obtain the Haralick glcm

% Here it is assumed that the diagonally opposite orientations are paired

% one after the other in the matrix

% If the above assumption is true with respect to the input glcm then

% setting the flag 'pairs' to 1 will compute the final glcms that would result

% by setting 'Symmetric' to true. If your glcm is computed using the

% Matlab version with 'Symmetric' flag you can set the flag 'pairs' to 0

%

% References:

% 1. R. M. Haralick, K. Shanmugam, and I. Dinstein, Textural Features of

% Image Classification, IEEE Transactions on Systems, Man and Cybernetics,

% vol. SMC-3, no. 6, Nov. 1973

% 2. L. Soh and C. Tsatsoulis, Texture Analysis of SAR Sea Ice Imagery

% Using Gray Level Co-Occurrence Matrices, IEEE Transactions on Geoscience

% and Remote Sensing, vol. 37, no. 2, March 1999.

% 3. D A. Clausi, An analysis of co-occurrence texture statistics as a

% function of grey level quantization, Can. J. Remote Sensing, vol. 28, no.

% 1, pp. 45-62, 2002

% 4. http://murphylab.web.cmu.edu/publications/boland/boland\_node26.html

%

%

% Example:

%

% Usage is similar to graycoprops() but needs extra parameter 'pairs' apart

% from the GLCM as input

% I = imread('circuit.tif');

% GLCM2 = graycomatrix(I,'Offset',[2 0;0 2]);

% stats = GLCM\_features1(GLCM2,0)

% The output is a structure containing all the parameters for the different

% GLCMs

%

% [Avinash Uppuluri: avinash\_uv@yahoo.com: Last modified: 11/20/08]

% If 'pairs' not entered: set pairs to 0

**if** **((**nargin **>** 2**)** **||** **(**nargin **==** 0**))**

error**(**'Too many or too few input arguments. Enter GLCM and pairs.'**);**

**elseif** **(** **(**nargin **==** 2**)** **)**

**if** **((**size**(**glcmin**,**1**)** **<=** 1**)** **||** **(**size**(**glcmin**,**2**)** **<=** 1**))**

error**(**'The GLCM should be a 2-D or 3-D matrix.'**);**

**elseif** **(** size**(**glcmin**,**1**)** **~=** size**(**glcmin**,**2**)** **)**

error**(**'Each GLCM should be square with NumLevels rows and NumLevels cols'**);**

**end**

**elseif** **(**nargin **==** 1**)** % only GLCM is entered

pairs **=** 0**;** % default is numbers and input 1 for percentage

**if** **((**size**(**glcmin**,**1**)** **<=** 1**)** **||** **(**size**(**glcmin**,**2**)** **<=** 1**))**

error**(**'The GLCM should be a 2-D or 3-D matrix.'**);**

**elseif** **(** size**(**glcmin**,**1**)** **~=** size**(**glcmin**,**2**)** **)**

error**(**'Each GLCM should be square with NumLevels rows and NumLevels cols'**);**

**end**

**end**

format long e

**if** **(**pairs **==** 1**)**

newn **=** 1**;**

**for** nglcm **=** 1**:**2**:**size**(**glcmin**,**3**)**

glcm**(:,:,**newn**)** **=** glcmin**(:,:,**nglcm**)** **+** glcmin**(:,:,**nglcm**+**1**);**

newn **=** newn **+** 1**;**

**end**

**elseif** **(**pairs **==** 0**)**

glcm **=** glcmin**;**

**end**

size\_glcm\_1 **=** size**(**glcm**,**1**);**

size\_glcm\_2 **=** size**(**glcm**,**2**);**

size\_glcm\_3 **=** size**(**glcm**,**3**);**

% checked

out**.**autoc **=** zeros**(**1**,**size\_glcm\_3**);** % Autocorrelation: [2]

out**.**contr **=** zeros**(**1**,**size\_glcm\_3**);** % Contrast: matlab/[1,2]

out**.**corrm **=** zeros**(**1**,**size\_glcm\_3**);** % Correlation: matlab

out**.**corrp **=** zeros**(**1**,**size\_glcm\_3**);** % Correlation: [1,2]

out**.**cprom **=** zeros**(**1**,**size\_glcm\_3**);** % Cluster Prominence: [2]

out**.**cshad **=** zeros**(**1**,**size\_glcm\_3**);** % Cluster Shade: [2]

out**.**dissi **=** zeros**(**1**,**size\_glcm\_3**);** % Dissimilarity: [2]

out**.**energ **=** zeros**(**1**,**size\_glcm\_3**);** % Energy: matlab / [1,2]

out**.**entro **=** zeros**(**1**,**size\_glcm\_3**);** % Entropy: [2]

out**.**homom **=** zeros**(**1**,**size\_glcm\_3**);** % Homogeneity: matlab

out**.**homop **=** zeros**(**1**,**size\_glcm\_3**);** % Homogeneity: [2]

out**.**maxpr **=** zeros**(**1**,**size\_glcm\_3**);** % Maximum probability: [2]

out**.**sosvh **=** zeros**(**1**,**size\_glcm\_3**);** % Sum of sqaures: Variance [1]

out**.**savgh **=** zeros**(**1**,**size\_glcm\_3**);** % Sum average [1]

out**.**svarh **=** zeros**(**1**,**size\_glcm\_3**);** % Sum variance [1]

out**.**senth **=** zeros**(**1**,**size\_glcm\_3**);** % Sum entropy [1]

out**.**dvarh **=** zeros**(**1**,**size\_glcm\_3**);** % Difference variance [4]

%out.dvarh2 = zeros(1,size\_glcm\_3); % Difference variance [1]

out**.**denth **=** zeros**(**1**,**size\_glcm\_3**);** % Difference entropy [1]

out**.**inf1h **=** zeros**(**1**,**size\_glcm\_3**);** % Information measure of correlation1 [1]

out**.**inf2h **=** zeros**(**1**,**size\_glcm\_3**);** % Informaiton measure of correlation2 [1]

%out.mxcch = zeros(1,size\_glcm\_3);% maximal correlation coefficient [1]

%out.invdc = zeros(1,size\_glcm\_3);% Inverse difference (INV) is homom [3]

out**.**indnc **=** zeros**(**1**,**size\_glcm\_3**);** % Inverse difference normalized (INN) [3]

out**.**idmnc **=** zeros**(**1**,**size\_glcm\_3**);** % Inverse difference moment normalized [3]

% correlation with alternate definition of u and s

%out.corrm2 = zeros(1,size\_glcm\_3); % Correlation: matlab

%out.corrp2 = zeros(1,size\_glcm\_3); % Correlation: [1,2]

glcm\_sum **=** zeros**(**size\_glcm\_3**,**1**);**

glcm\_mean **=** zeros**(**size\_glcm\_3**,**1**);**

glcm\_var **=** zeros**(**size\_glcm\_3**,**1**);**

% http://www.fp.ucalgary.ca/mhallbey/glcm\_mean.htm confuses the range of

% i and j used in calculating the means and standard deviations.

% As of now I am not sure if the range of i and j should be [1:Ng] or

% [0:Ng-1]. I am working on obtaining the values of mean and std that get

% the values of correlation that are provided by matlab.

u\_x **=** zeros**(**size\_glcm\_3**,**1**);**

u\_y **=** zeros**(**size\_glcm\_3**,**1**);**

s\_x **=** zeros**(**size\_glcm\_3**,**1**);**

s\_y **=** zeros**(**size\_glcm\_3**,**1**);**

% % alternate values of u and s

% u\_x2 = zeros(size\_glcm\_3,1);

% u\_y2 = zeros(size\_glcm\_3,1);

% s\_x2 = zeros(size\_glcm\_3,1);

% s\_y2 = zeros(size\_glcm\_3,1);

% checked p\_x p\_y p\_xplusy p\_xminusy

p\_x **=** zeros**(**size\_glcm\_1**,**size\_glcm\_3**);** % Ng x #glcms[1]

p\_y **=** zeros**(**size\_glcm\_2**,**size\_glcm\_3**);** % Ng x #glcms[1]

p\_xplusy **=** zeros**((**size\_glcm\_1**\***2 **-** 1**),**size\_glcm\_3**);** %[1]

p\_xminusy **=** zeros**((**size\_glcm\_1**),**size\_glcm\_3**);** %[1]

% checked hxy hxy1 hxy2 hx hy

hxy **=** zeros**(**size\_glcm\_3**,**1**);**

hxy1 **=** zeros**(**size\_glcm\_3**,**1**);**

hx **=** zeros**(**size\_glcm\_3**,**1**);**

hy **=** zeros**(**size\_glcm\_3**,**1**);**

hxy2 **=** zeros**(**size\_glcm\_3**,**1**);**

%Q = zeros(size(glcm));

**for** k **=** 1**:**size\_glcm\_3 % number glcms

glcm\_sum**(**k**)** **=** sum**(**sum**(**glcm**(:,:,**k**)));**

glcm**(:,:,**k**)** **=** glcm**(:,:,**k**)./**glcm\_sum**(**k**);** % Normalize each glcm

glcm\_mean**(**k**)** **=** mean2**(**glcm**(:,:,**k**));** % compute mean after norm

glcm\_var**(**k**)** **=** **(**std2**(**glcm**(:,:,**k**)))^**2**;**

**for** i **=** 1**:**size\_glcm\_1

**for** j **=** 1**:**size\_glcm\_2

out**.**contr**(**k**)** **=** out**.**contr**(**k**)** **+** **(**abs**(**i **-** j**))^**2.**\***glcm**(**i**,**j**,**k**);**

out**.**dissi**(**k**)** **=** out**.**dissi**(**k**)** **+** **(**abs**(**i **-** j**)\***glcm**(**i**,**j**,**k**));**

out**.**energ**(**k**)** **=** out**.**energ**(**k**)** **+** **(**glcm**(**i**,**j**,**k**).^**2**);**

out**.**entro**(**k**)** **=** out**.**entro**(**k**)** **-** **(**glcm**(**i**,**j**,**k**)\***log**(**glcm**(**i**,**j**,**k**)** **+** eps**));**

out**.**homom**(**k**)** **=** out**.**homom**(**k**)** **+** **(**glcm**(**i**,**j**,**k**)/(** 1 **+** abs**(**i**-**j**)** **));**

out**.**homop**(**k**)** **=** out**.**homop**(**k**)** **+** **(**glcm**(**i**,**j**,**k**)/(** 1 **+** **(**i **-** j**)^**2**));**

% [1] explains sum of squares variance with a mean value;

% the exact definition for mean has not been provided in

% the reference: I use the mean of the entire normalized glcm

out**.**sosvh**(**k**)** **=** out**.**sosvh**(**k**)** **+** glcm**(**i**,**j**,**k**)\*((**i **-** glcm\_mean**(**k**))^**2**);**

%out.invdc(k) = out.homom(k);

out**.**indnc**(**k**)** **=** out**.**indnc**(**k**)** **+** **(**glcm**(**i**,**j**,**k**)/(** 1 **+** **(**abs**(**i**-**j**)/**size\_glcm\_1**)** **));**

out**.**idmnc**(**k**)** **=** out**.**idmnc**(**k**)** **+** **(**glcm**(**i**,**j**,**k**)/(** 1 **+** **((**i **-** j**)/**size\_glcm\_1**)^**2**));**

u\_x**(**k**)** **=** u\_x**(**k**)** **+** **(**i**)\***glcm**(**i**,**j**,**k**);** % changed 10/26/08

u\_y**(**k**)** **=** u\_y**(**k**)** **+** **(**j**)\***glcm**(**i**,**j**,**k**);** % changed 10/26/08

% code requires that Nx = Ny

% the values of the grey levels range from 1 to (Ng)

**end**

**end**

out**.**maxpr**(**k**)** **=** max**(**max**(**glcm**(:,:,**k**)));**

**end**

% glcms have been normalized:

% The contrast has been computed for each glcm in the 3D matrix

% (tested) gives similar results to the matlab function

**for** k **=** 1**:**size\_glcm\_3

**for** i **=** 1**:**size\_glcm\_1

**for** j **=** 1**:**size\_glcm\_2

p\_x**(**i**,**k**)** **=** p\_x**(**i**,**k**)** **+** glcm**(**i**,**j**,**k**);**

p\_y**(**i**,**k**)** **=** p\_y**(**i**,**k**)** **+** glcm**(**j**,**i**,**k**);** % taking i for j and j for i

**if** **(**ismember**((**i **+** j**),[**2**:**2**\***size\_glcm\_1**]))**

p\_xplusy**((**i**+**j**)-**1**,**k**)** **=** p\_xplusy**((**i**+**j**)-**1**,**k**)** **+** glcm**(**i**,**j**,**k**);**

**end**

**if** **(**ismember**(**abs**(**i**-**j**),[**0**:(**size\_glcm\_1**-**1**)]))**

p\_xminusy**((**abs**(**i**-**j**))+**1**,**k**)** **=** p\_xminusy**((**abs**(**i**-**j**))+**1**,**k**)** **+**...

glcm**(**i**,**j**,**k**);**

**end**

**end**

**end**

% % consider u\_x and u\_y and s\_x and s\_y as means and standard deviations

% % of p\_x and p\_y

% u\_x2(k) = mean(p\_x(:,k));

% u\_y2(k) = mean(p\_y(:,k));

% s\_x2(k) = std(p\_x(:,k));

% s\_y2(k) = std(p\_y(:,k));

**end**

% marginal probabilities are now available [1]

% p\_xminusy has +1 in index for matlab (no 0 index)

% computing sum average, sum variance and sum entropy:

**for** k **=** 1**:(**size\_glcm\_3**)**

**for** i **=** 1**:(**2**\*(**size\_glcm\_1**)-**1**)**

out**.**savgh**(**k**)** **=** out**.**savgh**(**k**)** **+** **(**i**+**1**)\***p\_xplusy**(**i**,**k**);**

% the summation for savgh is for i from 2 to 2\*Ng hence (i+1)

out**.**senth**(**k**)** **=** out**.**senth**(**k**)** **-** **(**p\_xplusy**(**i**,**k**)\***log**(**p\_xplusy**(**i**,**k**)** **+** eps**));**

**end**

**end**

% compute sum variance with the help of sum entropy

**for** k **=** 1**:(**size\_glcm\_3**)**

**for** i **=** 1**:(**2**\*(**size\_glcm\_1**)-**1**)**

out**.**svarh**(**k**)** **=** out**.**svarh**(**k**)** **+** **(((**i**+**1**)** **-** out**.**senth**(**k**))^**2**)\***p\_xplusy**(**i**,**k**);**

% the summation for savgh is for i from 2 to 2\*Ng hence (i+1)

**end**

**end**

% compute difference variance, difference entropy,

**for** k **=** 1**:**size\_glcm\_3

% out.dvarh2(k) = var(p\_xminusy(:,k));

% but using the formula in

% http://murphylab.web.cmu.edu/publications/boland/boland\_node26.html

% we have for dvarh

**for** i **=** 0**:(**size\_glcm\_1**-**1**)**

out**.**denth**(**k**)** **=** out**.**denth**(**k**)** **-** **(**p\_xminusy**(**i**+**1**,**k**)\***log**(**p\_xminusy**(**i**+**1**,**k**)** **+** eps**));**

out**.**dvarh**(**k**)** **=** out**.**dvarh**(**k**)** **+** **(**i**^**2**)\***p\_xminusy**(**i**+**1**,**k**);**

**end**

**end**

% compute information measure of correlation(1,2) [1]

**for** k **=** 1**:**size\_glcm\_3

hxy**(**k**)** **=** out**.**entro**(**k**);**

**for** i **=** 1**:**size\_glcm\_1

**for** j **=** 1**:**size\_glcm\_2

hxy1**(**k**)** **=** hxy1**(**k**)** **-** **(**glcm**(**i**,**j**,**k**)\***log**(**p\_x**(**i**,**k**)\***p\_y**(**j**,**k**)** **+** eps**));**

hxy2**(**k**)** **=** hxy2**(**k**)** **-** **(**p\_x**(**i**,**k**)\***p\_y**(**j**,**k**)\***log**(**p\_x**(**i**,**k**)\***p\_y**(**j**,**k**)** **+** eps**));**

% for Qind = 1:(size\_glcm\_1)

% Q(i,j,k) = Q(i,j,k) +...

% ( glcm(i,Qind,k)\*glcm(j,Qind,k) / (p\_x(i,k)\*p\_y(Qind,k)) );

% end

**end**

hx**(**k**)** **=** hx**(**k**)** **-** **(**p\_x**(**i**,**k**)\***log**(**p\_x**(**i**,**k**)** **+** eps**));**

hy**(**k**)** **=** hy**(**k**)** **-** **(**p\_y**(**i**,**k**)\***log**(**p\_y**(**i**,**k**)** **+** eps**));**

**end**

out**.**inf1h**(**k**)** **=** **(** hxy**(**k**)** **-** hxy1**(**k**)** **)** **/** **(** max**([**hx**(**k**),**hy**(**k**)])** **);**

out**.**inf2h**(**k**)** **=** **(** 1 **-** exp**(** **-**2**\*(** hxy2**(**k**)** **-** hxy**(**k**)** **)** **)** **)^**0.5**;**

% eig\_Q(k,:) = eig(Q(:,:,k));

% sort\_eig(k,:)= sort(eig\_Q(k,:),'descend');

% out.mxcch(k) = sort\_eig(k,2)^0.5;

% The maximal correlation coefficient was not calculated due to

% computational instability

% http://murphylab.web.cmu.edu/publications/boland/boland\_node26.html

**end**

corm **=** zeros**(**size\_glcm\_3**,**1**);**

corp **=** zeros**(**size\_glcm\_3**,**1**);**

% using http://www.fp.ucalgary.ca/mhallbey/glcm\_variance.htm for s\_x s\_y

**for** k **=** 1**:**size\_glcm\_3

**for** i **=** 1**:**size\_glcm\_1

**for** j **=** 1**:**size\_glcm\_2

s\_x**(**k**)** **=** s\_x**(**k**)** **+** **(((**i**)** **-** u\_x**(**k**))^**2**)\***glcm**(**i**,**j**,**k**);**

s\_y**(**k**)** **=** s\_y**(**k**)** **+** **(((**j**)** **-** u\_y**(**k**))^**2**)\***glcm**(**i**,**j**,**k**);**

corp**(**k**)** **=** corp**(**k**)** **+** **((**i**)\*(**j**)\***glcm**(**i**,**j**,**k**));**

corm**(**k**)** **=** corm**(**k**)** **+** **(((**i**)** **-** u\_x**(**k**))\*((**j**)** **-** u\_y**(**k**))\***glcm**(**i**,**j**,**k**));**

out**.**cprom**(**k**)** **=** out**.**cprom**(**k**)** **+** **(((**i **+** j **-** u\_x**(**k**)** **-** u\_y**(**k**))^**4**)\***...

glcm**(**i**,**j**,**k**));**

out**.**cshad**(**k**)** **=** out**.**cshad**(**k**)** **+** **(((**i **+** j **-** u\_x**(**k**)** **-** u\_y**(**k**))^**3**)\***...

glcm**(**i**,**j**,**k**));**

**end**

**end**

% using http://www.fp.ucalgary.ca/mhallbey/glcm\_variance.htm for s\_x

% s\_y : This solves the difference in value of correlation and might be

% the right value of standard deviations required

% According to this website there is a typo in [2] which provides

% values of variance instead of the standard deviation hence a square

% root is required as done below:

s\_x**(**k**)** **=** s\_x**(**k**)** **^** 0.5**;**

s\_y**(**k**)** **=** s\_y**(**k**)** **^** 0.5**;**

out**.**autoc**(**k**)** **=** corp**(**k**);**

out**.**corrp**(**k**)** **=** **(**corp**(**k**)** **-** u\_x**(**k**)\***u\_y**(**k**))/(**s\_x**(**k**)\***s\_y**(**k**));**

out**.**corrm**(**k**)** **=** corm**(**k**)** **/** **(**s\_x**(**k**)\***s\_y**(**k**));**

% % alternate values of u and s

% out.corrp2(k) = (corp(k) - u\_x2(k)\*u\_y2(k))/(s\_x2(k)\*s\_y2(k));

% out.corrm2(k) = corm(k) / (s\_x2(k)\*s\_y2(k));

**end**

% Here the formula in the paper out.corrp and the formula in matlab

% out.corrm are equivalent as confirmed by the similar results obtained

% % The papers have a slightly different formular for Contrast

% % I have tested here to find this formula in the papers provides the

% % same results as the formula provided by the matlab function for

% % Contrast (Hence this part has been commented)

% out.contrp = zeros(size\_glcm\_3,1);

% contp = 0;

% Ng = size\_glcm\_1;

% for k = 1:size\_glcm\_3

% for n = 0:(Ng-1)

% for i = 1:Ng

% for j = 1:Ng

% if (abs(i-j) == n)

% contp = contp + glcm(i,j,k);

% end

% end

% end

% out.contrp(k) = out.contrp(k) + n^2\*contp;

% contp = 0;

% end

%

% end

% GLCM Features (Soh, 1999; Haralick, 1973; Clausi 2002)

% f1. Uniformity / Energy / Angular Second Moment (done)

% f2. Entropy (done)

% f3. Dissimilarity (done)

% f4. Contrast / Inertia (done)

% f5. Inverse difference

% f6. correlation

% f7. Homogeneity / Inverse difference moment

% f8. Autocorrelation

% f9. Cluster Shade

% f10. Cluster Prominence

% f11. Maximum probability

% f12. Sum of Squares

% f13. Sum Average

% f14. Sum Variance

% f15. Sum Entropy

% f16. Difference variance

% f17. Difference entropy

% f18. Information measures of correlation (1)

% f19. Information measures of correlation (2)

% f20. Maximal correlation coefficient

% f21. Inverse difference normalized (INN)

% f22. Inverse difference moment normalized (IDN)