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

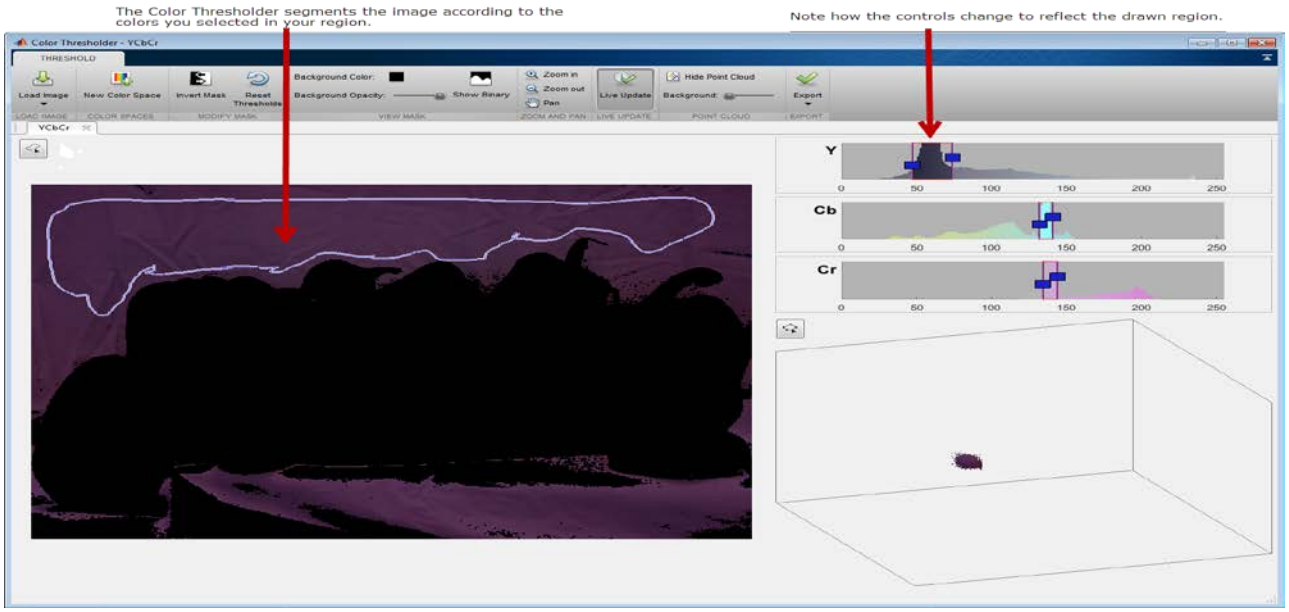
La fase di Segmentazione è stata tralasciata dal contesto di studio per la realizzazione di questo progetto, pertanto è stato possibile semplificare l’operazione concentrando la forza lavoro del team per lo studio e la ricerca delle features.

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

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

In 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.

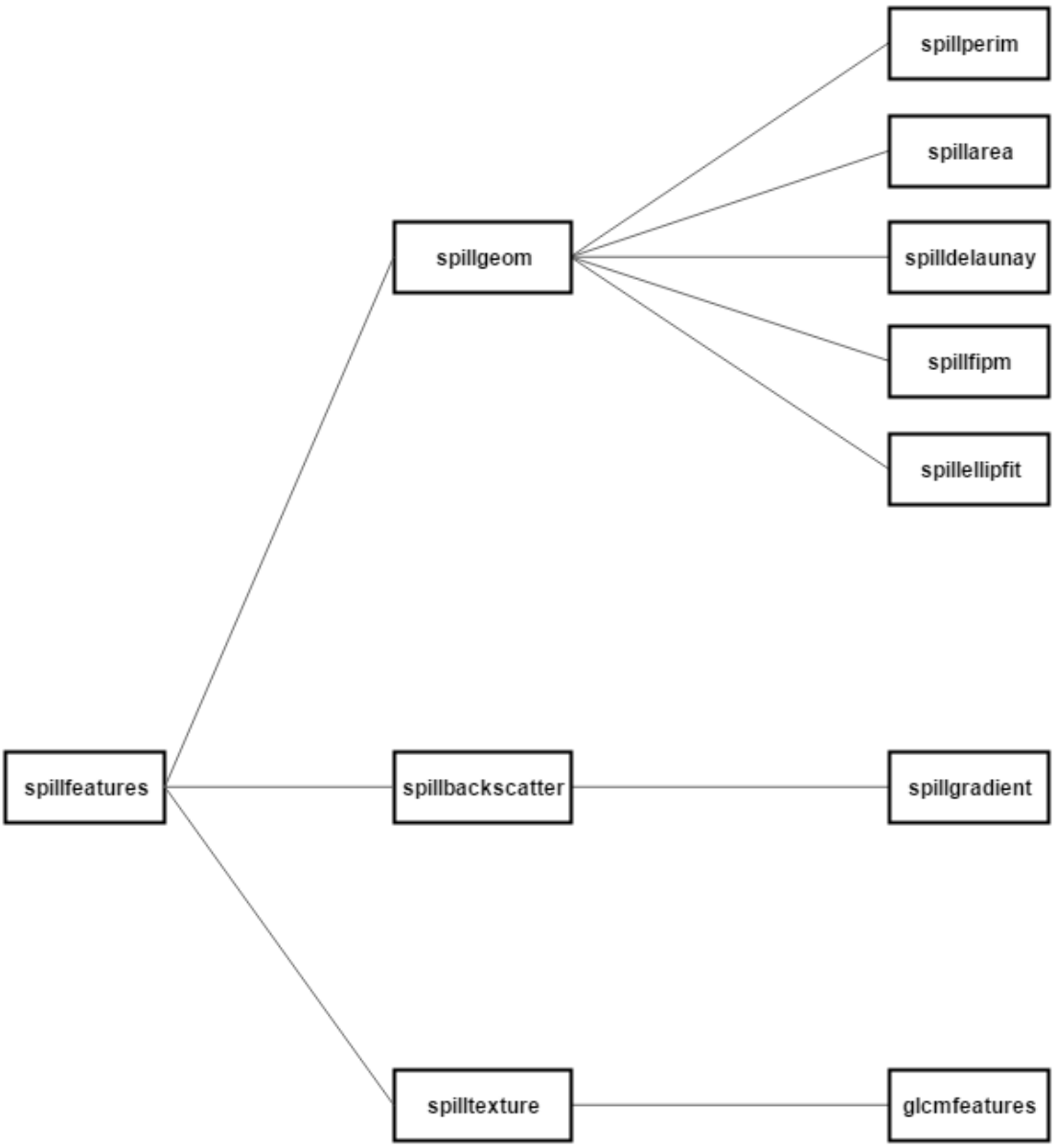
Tramite l’utilizzo di questo toolbox, è stato possibile, adoperando delle operazioni manuali di selezione delle aree dello sfondo di segmentare l’immagine, nel nostro caso la macchia di petrolio.



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.



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.

```
>> spillbegin
>> f = spillfeatures( gspill, gback, s )

f =

    Geometrical: [1x1 struct]
    Backscatter: [1x1 struct]
        Texture: [1x1 struct]

>> f.Geometrical

ans =

     Perimeter: 11070
          Area: 1.617588750000000e+06
    Complexity: 2.455322988115242e+00
        Length: 3.968644639614441e+03
          FIPM: 5.272670350224763e-03
            EL: 2.209119555287649e+03
            EW: 1.154700538379251e+00
            EA: 9.994773028306162e-01
```

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

Complexity (C), defined as

$$C := \frac{P}{2\sqrt{\pi A}}.$$

This feature will take a small numerical value for regions with simple geometry and larger values for complex geometrical regions.

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.

- Length (L): sum of skeleton edges (obtained by Delaunay triangulation), that build the main line.
- Width (W): mean value of Delaunay triangles which are crossed by main line.
- Length To Width Ratio (LWR), defined as

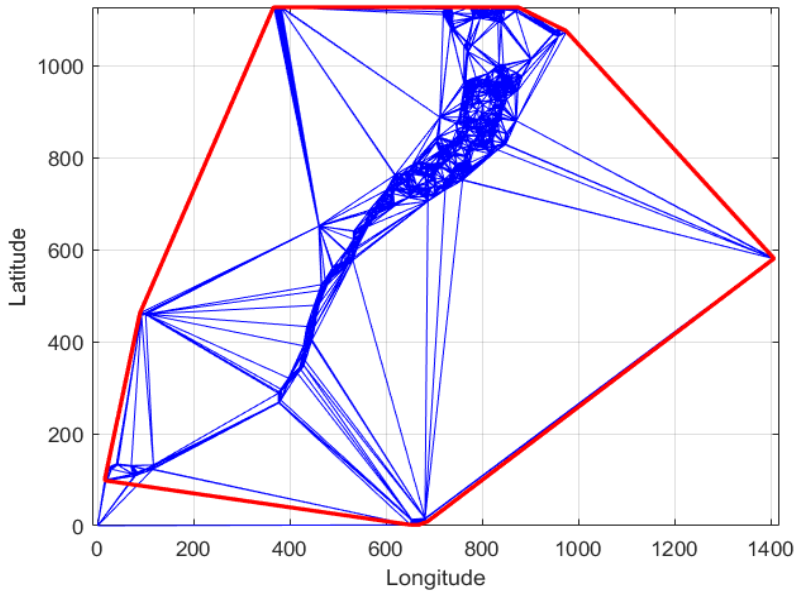
$$LWR := \frac{L}{W}.$$

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

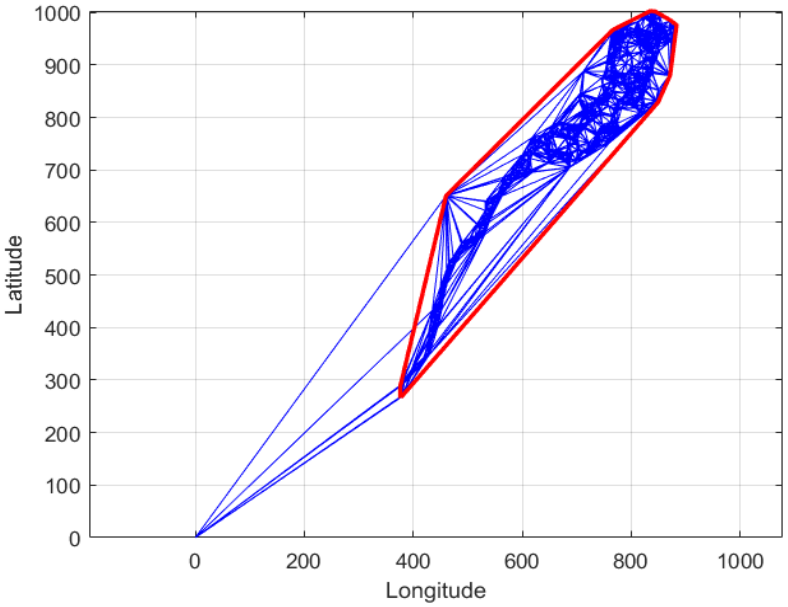
Al contrario siamo riusciti ad estrarre il parametro Length in questo modo:

```
%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 d'olio. 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:

$$\mathrm{FIPM} := \frac{1}{n^2} \sum_{i=1}^{n} \left[(x_i - x_c)^2 + (y_i - y_c)^2 \right]$$

with

$$x_c := \frac{1}{n} \sum_{i=1}^{n} x_i \ , \ y_c = \frac{1}{n} \sum_{i=1}^{n} y_i,$$

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

- Ellipse-Length (EL): value of main axe of an ellipse fitted to the data.
- Ellipse-Width (EW): value of minor axe of an ellipse fitted to the data.
- Ellipse-Asymetry (EA), defined as

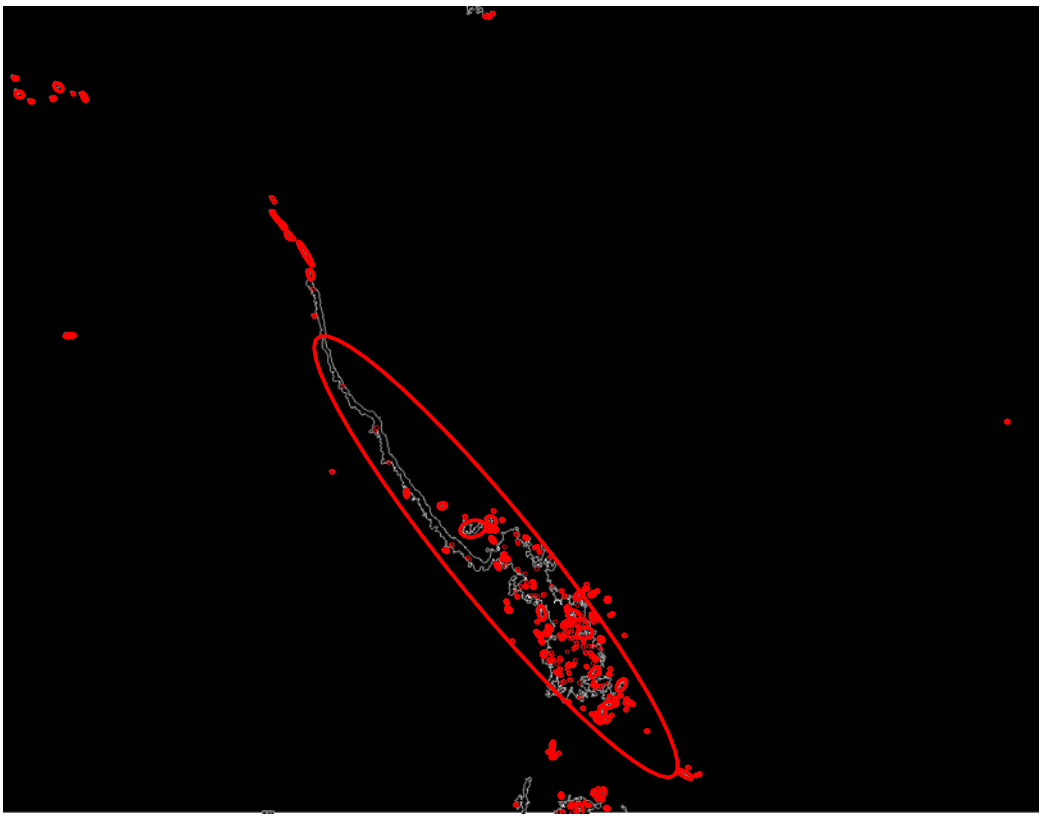
$$\mathrm{EA} := 1 - \frac{\mathrm{EW}}{\mathrm{EL}}.$$

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 鑓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

X. GMax, GMe e GSt

Questi parametri fanno parte delle Backscatters features, rappresentano le ultime tre…. DA RICOLLOCARE

Queste tre metriche rappresentano:

- **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.

E sono calcolate adoperando la funzione [gMax, gMe, gSt]= spillgradient(img).

Questa funzione 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 dev standard restiduendo i valori in dB.

```
Calcolo del Gradient Magnitude e la direzione usando sobel (default, Gx e Gy,
vengono calcolati dentro imgradient se non presenti usando la funzione [Gx, Gy] = imgradientxy(I);
[Gmag, Gdir] = imgradient(I, 'sobel'); %Gmag e Gdir sono due matrici della stessa dimensione
dell'immagine originale, contenenti rispettivamente
 [Max,k]=max(max(Gmag)); %individuo il valore massimo

% effettua la media dei valori degli edge che fanno parte del bordo.

[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);
```