Segmentazione a soglia e Segmentazione Marker-Controlled applicate alla Diagnosi dei Tumori del Polmone

Salvatore Trisciuoglio

Simone Carmine Ciavarella

1. Introduzione 3
   - Riferimenti 3
2. Stato Dell'Arte 4
3. Scelta delle tecniche 5
4. Descrizione Tecniche 5
   1. Operazioni Morfologiche 5
   2. Threshold Segmentation 7
   3. Segmentazione Marker Controlled 9
5. Implementazione 12
   1. Metodi Comuni 12
      - MRI Import 12
      - Automazione calcolo performance 12
   2. Threshold Segmentation 19
   3. Marker-Controlled Segmentation 21
6. Dataset & Performance 24
   4. Performance Threshold Segmentation 25
   5. Performance Marker Controlled Segmentation 28
7. Conclusione 32
8. Bibliografia 33

1. Introduzione

È stato fornito un dataset composto da file *.nii.gz* contenenti risonanze magnetiche effettuate su polmoni affetti da tumore.

Questo progetto punta ad analizzare i dati forniti e, tramite l'utilizzo di operazioni di image processing e delle tecniche di segmentazione, individuare i tumori.

- Riferimenti

Nome	URL
Dataset - *Task06_Lung.tar*	MSD - Google Drive
Repository GitHub progetto	Repository

2. Stato Dell'Arte

La segmentazione e la rilevazione dei tumori polmonari nelle immagini di risonanza magnetica (MRI) rappresentano un campo di ricerca cruciale nell'elaborazione delle immagini mediche. Le tecniche avanzate di segmentazione automatica mirano a migliorare la diagnosi precoce, ridurre i costi medici e aumentare le probabilità di successo del trattamento. Di seguito una panoramica dello stato dell'arte delle tecniche di segmentazione delle immagini utilizzate per identificare i tumori polmonari nelle immagini MRI [4].

- **Segmentazione a soglia**: è una tecnica semplice ed efficace che si basa sull'identificazione di valori di intensità specifici per separare le strutture di interesse dallo sfondo.
- **Region-based segmentation**: partendo da uno o più "pixel seme", gli algoritmi di crescita delle regioni raggruppano insieme i pixel vicini con caratteristiche simili. Gli algoritmi possono essere agglomerativi o divisivi.
- **Segmentazione marker controlled**: questa tecnica utilizza markers (etichette iniziali) per guidare il processo di segmentazione. Questa tecnica è utile per segmentare strutture complesse come i tumori, offrendo un alto grado di precisione e robustezza.
- **Edge Detection**: i metodi di rilevamento dei bordi identificano i confini degli oggetti o delle classi rilevando le discontinuità nella luminosità o nel contrasto.
- **Segmentazione basata sul clustering**: un metodo di apprendimento non supervisionato, gli algoritmi di clustering dividono i dati visivi in cluster di pixel con valori simili. Una variante comune è il clustering K-means, in cui k è il numero di cluster. I valori dei pixel sono tracciati come punti di dati e k punti casuali sono selezionati come centro di un cluster (centroide). Ogni pixel è assegnato a un cluster in base al centroide più vicino. I centroidi vengono quindi ricollocati alla media di ciascun cluster e il processo viene ripetuto, ricollocando i centroidi ad ogni iterazione fino a quando i cluster non si sono stabilizzati.

Le tecniche di segmentazione automatica sono state applicate con successo nella rilevazione di tumori polmonari in varie ricerche. Studi clinici hanno dimostrato che l'uso di algoritmi avanzati migliora l'accuratezza della diagnosi e la pianificazione del trattamento. L'integrazione di tecniche di pre-processing con metodi di segmentazione aumenta ulteriormente la precisione della rilevazione.

3. Scelta delle tecniche

La **segmentazione** di un'immagine nell'elaborazione digitale delle immagini è il processo di partizione di un'immagine in regioni significative. L' immagine digitale viene suddivisa in insiemi di pixel. Lo scopo della segmentazione è semplificare e/o cambiare la rappresentazione delle immagini in qualcosa che è più significativo e facile da analizzare [1].

La segmentazione è di solito utilizzata per localizzare oggetti e bordi. Più precisamente, la segmentazione è il processo con il quale si classificano i pixel dell'immagine che hanno caratteristiche comuni, pertanto ciascun pixel in una regione è simile agli altri della stessa regione per una qualche proprietà o caratteristica (colore, intensità o texture). Regioni adiacenti sono significativamente differenti rispetto ad almeno una di queste caratteristiche.

Sono state implementate due tecniche di segmentazione: *Threshold Segmentation* e *Marker-Controlled Segmentation.*

4. Descrizione Tecniche

Di seguito verranno descritti i concetti teorici alla base della segmentazione basata su soglia e della segmentazione marker-controlled.

In entrambi gli algoritmi è stato fatto uso di operazioni morfologiche in modo da facilitare l'eliminazione di rumore e il rilevamento di aree di interesse.

1. Operazioni Morfologiche

Le operazioni morfologiche di opening-closing (apertura-chiusura) e opening-closing by reconstruction (apertura-chiusura per ricostruzione) sono utilizzate per rimuovere rumore, isolare strutture specifiche e mantenere le caratteristiche importanti di un'immagine. Ci sono differenze significative tra queste due tecniche in termini di risultati e modalità operative.

- **Apertura (Opening)**

  Definizione: Un'erosione seguita da una dilatazione.

  Scopo: Rimuovere piccoli oggetti e rumore senza modificare troppo le forme più grandi.

  Effetto: Le piccole strutture e il rumore vengono rimossi, mentre gli oggetti più grandi possono essere leggermente ridotti in dimensione.

- **Chiusura (Closing)**

  Definizione: Una dilatazione seguita da un'erosione.

  Scopo: Riempire piccoli buchi e collegare stretti spazi tra gli oggetti.

  Effetto: Le piccole lacune e i piccoli spazi tra gli oggetti vengono riempiti, mentre le strutture più grandi possono essere leggermente ingrandite.

- **Apertura-Chiusura (Opening-Closing)**

  Sequenza: Esecuzione dell'apertura seguita dalla chiusura.

  Scopo: Rimuovere rumore e piccole strutture senza modificare significativamente le forme più grandi.

  Effetto: La combinazione di apertura e chiusura rimuove piccole strutture e rumore, e riempie piccoli buchi, mantenendo una migliore rappresentazione delle strutture più grandi rispetto a fare solo una delle operazioni.

- **Apertura per Ricostruzione (Opening by Reconstruction)**

  Definizione: Un'erosione seguita da una ricostruzione morfologica con l'immagine originale.

  Scopo: Preservare meglio le strutture originali rispetto all'apertura tradizionale.

  Effetto: Le piccole strutture e il rumore vengono rimossi, ma le forme più grandi rimangono intatte, senza essere ridotte in dimensione.

- **Chiusura per Ricostruzione (Closing by Reconstruction)**

  Definizione: Una dilatazione seguita da una ricostruzione morfologica con l'immagine originale.

  Scopo: Preservare meglio le strutture originali rispetto alla chiusura tradizionale.

  Effetto: Le piccole lacune e i piccoli spazi vengono riempiti, ma le forme più grandi rimangono intatte, senza essere ingrandite.

- **Apertura-Chiusura per Ricostruzione (Opening-Closing by Reconstruction)**

  Sequenza: Esecuzione dell'apertura per ricostruzione seguita dalla chiusura per ricostruzione.

  Scopo: Rimuovere rumore e piccole strutture preservando al massimo le caratteristiche delle forme originali.

Effetto: Le piccole strutture e il rumore vengono rimossi, e i piccoli buchi vengono riempiti, senza modificare significativamente le dimensioni o la forma delle strutture più grandi.

2. Threshold Segmentation

La segmentazione a soglia trasforma un'immagine in scala di grigi in un'immagine binaria, dove i pixel sono assegnati a due categorie:

1. **Foreground (primo piano)**: pixel la cui intensità è superiore al valore soglia.
2. **Background (sfondo)**: pixel la cui intensità è inferiore o uguale al valore soglia.

La segmentazione a soglia è composta dai seguenti passi:

1. **Immagine in Scala di Grigi**: Il primo passo è convertire l'immagine originale in un'immagine in scala di grigi se non lo è già. Questo è necessario perché la segmentazione a soglia lavora sull'intensità dei pixel, che è più semplice da gestire in scala di grigi.
2. **Determinazione del Valore Soglia**: Il valore soglia può essere scelto manualmente o determinato automaticamente tramite algoritmi specifici. Uno degli algoritmi più popolari per determinare automaticamente il valore soglia ottimale è il metodo di Otsu. Questo metodo calcola il valore soglia che minimizza la varianza intra-classe, cioè la differenza di intensità all'interno delle classi del primo piano e dello sfondo.
3. **Applicazione del Valore Soglia**: Una volta determinato il valore soglia, si procede a classificare ogni pixel dell'immagine in scala di grigi. Se l'intensità di un pixel è superiore al valore soglia, viene classificato come pixel di primo piano e impostato a bianco (1); altrimenti, viene classificato come pixel di sfondo e impostato a nero (0).

Per applicare al meglio la segmentazione a soglia bisogna seguire le seguenti fasi:

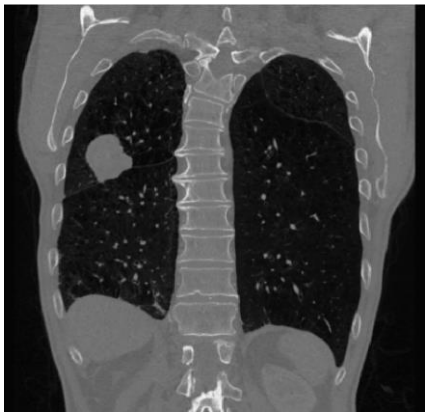

Figura 1 Immagine di input

1. **Preparazione dell'Immagine**:
    - Acquisire l'immagine e convertirla in scala di grigi.

- Utilizzare tecniche di pre-elaborazione come il filtraggio per ridurre il rumore, che può influenzare negativamente la segmentazione.

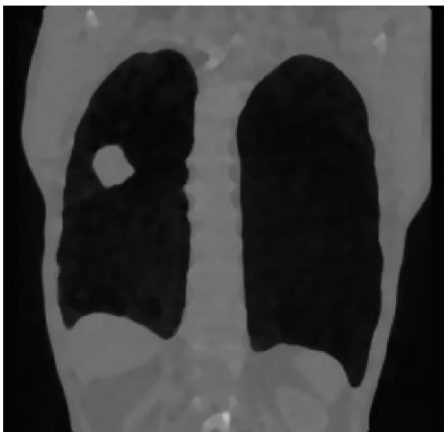

Figura 2 Immagine convertita in scala di grigi e filtrata

2. **Calcolo del Valore Soglia**:
    - *Selezione Manuale*: Osservare l'istogramma dell'immagine per scegliere un valore soglia appropriato.
    - *Selezione Automatica*: Applicare il metodo di Otsu o altri algoritmi automatici per determinare il valore soglia ottimale.

3. **Segmentazione:**
    - Applicare il valore soglia all'immagine in scala di grigi per ottenere un'immagine binaria.

4. **Sottrazione dell'immagine**
    - *Area Opening*: Questa tecnica viene utilizzata per rimuovere i componenti indesiderati da un'immagine, serve a rimuovere da un'immagine binaria tutti i componenti connessi che hanno il numero di pixel inferiori a un valore impostato. Nel nostro caso utilizzeremo L'Area Opening per rimuovere il tumore dall'immagine
    - *Sottrazione dell'immagine*: Sottraendo l'immagine senza tumore all'immagine segmentata otteniamo un'immagine contenente il solo tumore

Figura 3 Immagine segmentata

Figura 4 Immagine dopo l'applicazione dell'Area Opening

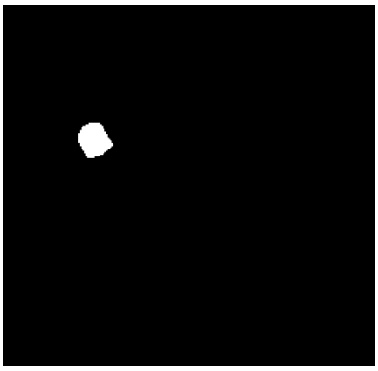

Figura 5 Risultato della sottrazione dell'immagine

3. Segmentazione Marker Controlled

La segmentazione **watershed** è una tecnica di elaborazione delle immagini utilizzata per segmentare un'immagine in regioni basate sulla topologia dei suoi livelli di grigio (i pixel chiari sono alti e i pixel scuri sono bassi). Il nome watershed (spartiacque) deriva dall'analogia con il processo di separazione delle aree drenate da diverse sorgenti d'acqua. Avremo *bacini idrografici* (catchment basins) e *linee di cresta spartiacque* (watershed ridge lines).

Di seguito i passaggi teorici della segmentazione watershed:

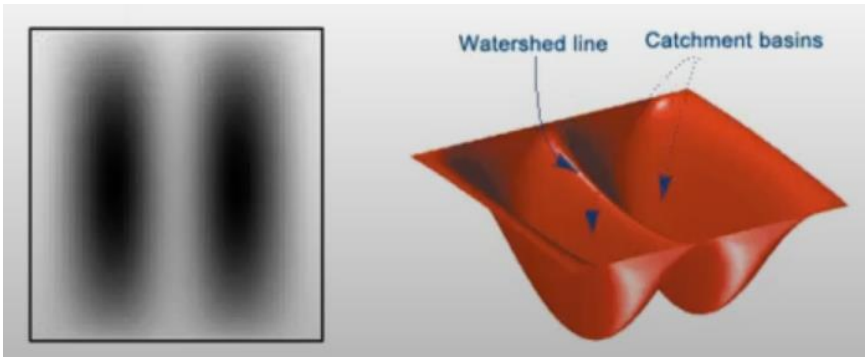

Figura 3 Immagine greyscale che rappresenta neri i pixel bassi e bianchi quelli alti

La segmentazione Marker-controlled watershed segue questa procedura:

- Conversione immagine in scala di grigi
- Pre-processing dell'immagine utilizzano tecniche morfologiche
- Calcolare la grandezza del gradiente (*gradient magnitude*), Figura 4. Il gradiente risulta più alto sui bordi degli oggetti e basso dentro agli oggetti (ai bordi degli oggetti si presuppone una differenza di intensità di colore maggiore).

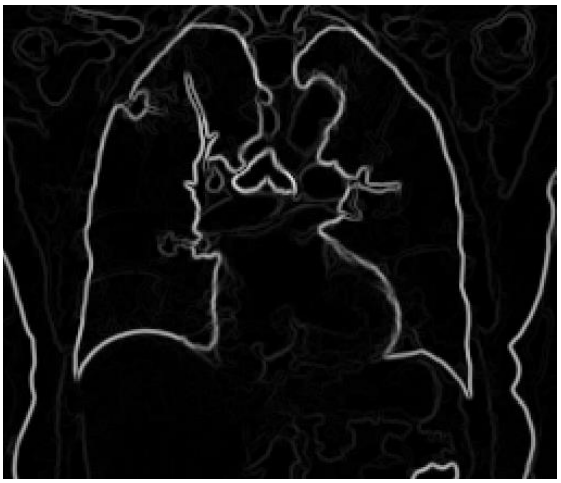

Figura 4 Gradient Magnitude calcolata su una immagine del dataset

Applicare la trasformata watershed direttamente sull'immagine del gradiente implicherebbe *oversegmentation* (Figura 5)

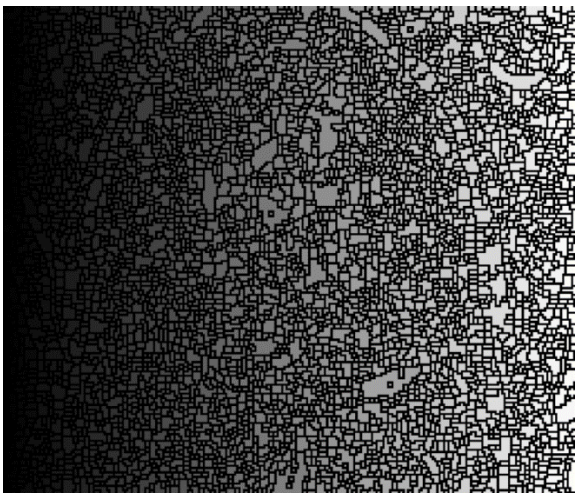

Figura 5 Oversegmentation ottenuta tramite il calcolo della trasformata watershed sul polmone. Non sono stati utilizzati marcatori.

- **Inondazione da Punti di Minima**: trattare l'immagine come un paesaggio tridimensionale dove le intensità dei pixel rappresentano l'altitudine. Iniziare a "inondare" il paesaggio da ogni punto di minima locale (cioè, i punti più bassi dell'immagine). Questi punti saranno le sorgenti dell'acqua.
- **Espansione delle Regioni Inondate**: Continuare l'inondazione facendo espandere l'acqua da ciascuna sorgente. Le regioni si espandono fino a incontrare i bordi (i gradienti elevati) o altre regioni in espansione.
- **Creazione delle Watershed Ridge Lines**: quando due regioni in espansione si incontrano, viene creata una linea di watershed che separa le regioni. Queste linee rappresentano i confini tra le diverse regioni segmentate.
- **Assegnazione delle Etichette tramite watershed function**: assegnare un'etichetta unica a ciascuna regione delimitata dalle linee di watershed.

L'approccio **marker-controlled** introduce dei *marcatori* per migliorare la segmentazione e superare problemi come l'oversegmentation (troppi segmenti) o l'undersegmentation (pochi segmenti).

I **marcatori interni** sono pixel o regioni che si sa con certezza appartenere agli oggetti di interesse. Questi marcatori vengono definiti manualmente o automaticamente, spesso basandosi su caratteristiche come l'intensità dell'immagine.

I **marcatori esterni** sono pixel o regioni che si sa con certezza appartenere allo sfondo dell'immagine. Anche questi vengono definiti manualmente o automaticamente, spesso utilizzando tecniche di distanza dai marcatori interni.

Esempio: I marcatori esterni possono essere definiti come i pixel che si trovano a una certa distanza dai marcatori interni, spesso utilizzando la trasformata della distanza.

Una volta create marcatori interni ed esterni bisogna modificare la mappa di gradiente dell'immagine originale imponendo i marcatori. Questo si traduce in un'immagine in cui i marcatori vengono usati per "segnare" le regioni da cui iniziare l'inondazione. Infine, applicare l'algoritmo di watershed.

Si avrà un'immagine segmentata in cui le regioni sono ben definite grazie all'uso dei marcatori.

Vantaggi dell'Approccio Marker-Controlled:

- ✓ Maggiore Precisione: I marcatori aiutano a ottenere una segmentazione più accurata.
- ✓ Riduzione dell'Oversgementation: Limita il numero di segmenti non desiderati.
- ✓ Flessibilità: Permette di adattare la segmentazione a specifiche caratteristiche dell'immagine.

5. Implementazione

Questo capitolo rappresenterà gli algoritmi utilizzati, opportunamente commentati per fini esplicativi accademici.

1. Metodi Comuni

Nel seguito verranno descritte alcune logiche, sviluppate per automatizzare fasi del lavoro, con il fine di velocizzare i processi di valutazione del codice e dei parametri.

➢ MRI Import

Sono stati utilizzati file con estensione .nii.gz (che è possible aprire con programmi come Slicer [5]). Questi file contengono i frame, nelle tre dimensioni spaziali, di risonanze magnetiche ai polmoni.

1. È stata estratta la coordinata, che corrisponde alla massima estensione tumorale, da ogni file contenente etichetta di verità. Viene salvata la verità in un path specifico.
2. È stata ricercata la coordinata, trovata al punto precedente, nel file della risonanza magnetica rappresentante i polmoni. Viene salvata l'immagine in un path specifico.

Si avrà a disposizione una folder structure che raccoglierà ogni campione in una cartella e li fornirà agli algoritmi di Threshold Segmentation e Marker-Controlled Segmentation. Questi ultimi salveranno coerentemente i risultati (sovrapposizioni della input image con la rilevazione) e le rilevazioni (tramite matrice logica).

```
clear;
close all;

Helpers.datasetfolderstructuring('Task06_Lung/imagesTr', 'Task06_Lung/labelsTr');
```

Figura 6 Chiamata alla funzionalità MRI Import

➢ Automazione calcolo performance

Per calcolare le performance è stato scritto il codice in *Evaluation.m*. Questo cicla le immagini delle occorrenze tumorali trovate dalla Threshold Segmentation, dalla Marker Controlled Segmentation e le relative immagini di verità. Una volta acquisite le immagini le ridimensiona ad una risoluzione standard e, paragonando i pixel, calcola in automatico: la matrice di confusione, l'Accuracy, il True Positive Rate e il True Negative Rate per entrambi i metodi. Infine, stampa i risultati in console.

```
clear;
close all;

% Carica i percorsi delle immagini MRI del polmone
examplesFolders = Helpers.elenca_file_con_prefisso('Dataset', 'lung');

newSize = [256, 256];

TPThreshold = 0;
TNThreshold = 0;
FPThreshold = 0;
FNThreshold = 0;

TPMarkerControlled = 0;
TNMarkerControlled = 0;
FPMarkerControlled = 0;
FNMarkerControlled = 0;

% Cicla tutte le immagini di test
for j = 1 : size(examplesFolders,2)

    folderToSave = fullfile('Dataset', examplesFolders(j));
    thresholdOccurences = imread(char(fullfile(folderToSave,
'soglia_occorrenze_rilevate.png')));
    markerControlledOccurences = imread(char(fullfile(folderToSave,
'marker_controlled_occorrenze_rilevate.png')));
    labels = imread(char(fullfile(folderToSave, 'labelImage.png')));

    % Effettua il resize delle immagini per standardizzarle
    thresholdOccurences = imresize(thresholdOccurences, newSize);
    markerControlledOccurences = imresize(markerControlledOccurences, newSize);
    labels = imresize(labels, newSize);

    % Calcola le matrici di confusione per il Threshold
    [TP, TN, FP, FN] = Helpers.calculate_confusion_matrix(labels, thresholdOccurences);

    % Somma i valori delle matrici di confusione
    TPThreshold = TPThreshold + TP;
    TNThreshold = TNThreshold + TN;
    FPThreshold = FPThreshold + FP;
    FNThreshold = FNThreshold + FN;

    % Calcola le matrici di confusione per il Marker Controlled
    [TPa, TNa, FPa, FNa] = Helpers.calculate_confusion_matrix(labels,
markerControlledOccurences);

    % Somma i valori delle matrici di confusione
    TPMarkerControlled = TPMarkerControlled + TPa;
    TNMarkerControlled = TNMarkerControlled + TNa;
    FPMarkerControlled = FPMarkerControlled + FPa;
    FNMarkerControlled = FNMarkerControlled + FNa;
end

% Calcola Accuracy, True Positive Rate e True Negative Rate per il
% Threshold
ACCThreshold = (TPThreshold + TNThreshold) / (TPThreshold + TNThreshold + FPThreshold +
FNThreshold);
TPRThreshold = TPThreshold / (TPThreshold + FNThreshold);
TNRThreshold = TNThreshold / (TNThreshold + FPThreshold);

% Calcola Accuracy, True Positive Rate e True Negative Rate per il
```

```
% Marker Controlled
ACCMarkerControlled = (TPMarkerControlled + TNMarkerControlled) / (TPMarkerControlled +
TNMarkerControlled + FPMarkerControlled + FNMarkerControlled);
TPRMarkerControlled = TPMarkerControlled / (TPMarkerControlled + FNMarkerControlled);
TNRMarkerControlled = TNMarkerControlled / (TNMarkerControlled + FPMarkerControlled);

% Stampa dei risultati
fprintf('-----THRESHOLD-----\n');

fprintf('TPThreshold = %d\n', TPThreshold);
fprintf('TNThreshold = %d\n', TNThreshold);
fprintf('FPThreshold = %d\n', FPThreshold);
fprintf('FNThreshold = %d\n', FNThreshold);

fprintf('ACCThreshold = %.4f\n', ACCThreshold);
fprintf('TPRThreshold = %.4f\n', TPRThreshold);
fprintf('TNRThreshold = %.4f\n', TNRThreshold);

fprintf('-----MARKER CONTROLLED-----\n');

fprintf('TPMarkerControlled = %d\n', TPMarkerControlled);
fprintf('TNMarkerControlled = %d\n', TNMarkerControlled);
fprintf('FPMarkerControlled = %d\n', FPMarkerControlled);
fprintf('FNMarkerControlled = %d\n', FNMarkerControlled);

fprintf('ACCMarkerControlled = %.4f\n', ACCMarkerControlled);
fprintf('TPRMarkerControlled = %.4f\n', TPRMarkerControlled);
fprintf('TNRMarkerControlled = %.4f\n', TNRMarkerControlled);
```

Di seguito viene inserita per intero la classe Helpers.m che contiene metodi comuni utilizzati nelle altre parti della codebase.

```
classdef Helpers

    methods(Static)

        % Crea il dataset di immagini partendo dai file delle Mri
        % selezionando dal piano orizzontale le immagini con la dimensione
        % del tumore più estesa
        function datasetfolderstructuring(mrisTrainingPath, mrisLabelPath)
            baseFolder = 'Dataset';
            % Creazione della cartella
            if ~exist(baseFolder, 'dir')
                mkdir(baseFolder);
            end

            % Carica i percorsi delle immagini MRI del polmone
            trainingFileNames = Helpers.elenca_file_con_prefisso(mrisTrainingPath,
'lung');

            for i = 1 : size(trainingFileNames,2)
                exampleName = char(strtok(trainingFileNames(i), "."));
                exampleFolderName = fullfile(baseFolder, exampleName);
                if ~exist(exampleFolderName, 'dir')
                    mkdir(exampleFolderName);
                end

                [trainingImage,labelImage] = Helpers.calculate_images_for_trai-
ning_and_label(fullfile(mrisTrainingPath, char(trainingFileNames(i))), ...
                    fullfile(mrisLabelPath , char(trainingFileNames(i))));

                if ~exist(exampleFolderName, 'dir')
                    mkdir(exampleFolderName);
                else
                    rmdir(exampleFolderName, 's');
                    mkdir(exampleFolderName);
                end

                imwrite(trainingImage, fullfile(baseFolder, exampleName ,'trainingI-
mage.png'));
                imwrite(labelImage, fullfile(baseFolder, exampleName ,'labelI-
mage.png'));
            end

        end

        % Carica i percorsi delle immagini MRI del polmone
        function fileNames = elenca_file_con_prefisso(folderPath, prefisso)
             % Ottieni una lista di tutti i file nella cartella specificata
             listaFiles = dir(fullfile(folderPath, [prefisso, '*']));

             % Estrai i nomi dei file dalla struttura 'listaFiles'
             fileNames = {listaFiles.name};
        end

        % Utilizzando i percorsi delle mri relative al training ed alla
        % label trova le immagini con la dimensione del tumore massima: la
        % estrae, la normalizza (range 0-1) e la restituisce come output
        function [imageTraining, imageLabel] = calculate_images_for_training_and_la-
bel(imagePath, labelPath)
            % Calcoliamo la nuova proporzione in base ai dati niftiread
            label = niftiread(labelPath);
            [imageLabel, depth] = Helpers.search_max_region_in_label(label);
```

```
            imageTraining = niftiread(imagePath);
            infoTraining = niftiinfo(imagePath);
            imageTraining = Helpers.extractyimage(imageTraining, depth);

            imageTraining = imrotate(Helpers.mriproportionadjustement(imageTraining,
infoTraining), 90);
            imageTraining = Helpers.imagenormalize(imageTraining);

            infoLabel = niftiinfo(labelPath);

            imageLabel = imrotate(Helpers.mriproportionadjustement(imageLabel,
infoLabel), 90);
        end

        % Corregge la proporzione dell'immagine importata adattandola alla
        % pixel dimension degli assi di interesse.
        % Input: immagine e info niftiread .nii.gz
        function image_corrected = mriproportionadjustement(image, info)
            x = info.ImageSize(1) * info.PixelDimensions(1);
            y = info.ImageSize(3) * info.PixelDimensions(3);
            new_size = [x, y];

            image_corrected = imresize(image, new_size);
        end

        % Proporziona l'intensità dei pixel in un range tra 0 e 1
        function img_normalized = imagenormalize(image)
            min_val = min(image(:)); % Trova il valore minimo dei pixel
            max_val = max(image(:)); % Trova il valore massimo dei pixel
            img_normalized = (double(image) - min_val) / (max_val - min_val); %
Normalizza l'immagine
        end

        % Ritorna una immagine proporzionata in altezza rispetto alla
        % larghezza obiettivo fornita
        function imageResized = resize(img, newWidth)
            % Calcoliamo la nuova altezza proporzionale b:h=B:H
            originalSize = size(img);
            newHeight = round(newWidth * originalSize(1) / originalSize(2));

            % Ridimensionamento dell'immagine
            new_size = [newHeight, newWidth];
            imageResized = imresize(img, new_size);
        end

        % Trasforma l'immagine in scala di grigi (se necessario) e,
        % utilizzando il metodo graythresh, calcola la soglia che poi
        % utilizza per binarizzare l'immagine
        function binImage = otsubin(image)
            image = Helpers.rgb2gray(image);
            greyLevel = graythresh(image);
            binImage = imbinarize(image, greyLevel);
        end

        % Data una MRI di una label trova il frame con l'area maggiore del tumore
        % Output: l'immagine con la maggior estensione tumorale e la sua profondità
        function [image, depth] = search_max_region_in_label(mri)
            depth = 0;
            maxArea = 0;
            for i = 1 : size(mri, 2)
                extractedImage = Helpers.extractyimage(mri, i);
```

```
            extractedImageBin = Helpers.otsubin(extractedImage);
            [regions, num_labels] = bwlabel(extractedImageBin);

            props = regionprops(regions, 'Area');
            for j = 1 : num_labels
                if props(j).Area > maxArea
                    maxArea = props(j).Area;
                    depth = i;
                    image = extractedImageBin;
                end
            end
        end

    end

    % Data una MRI ed una profondità estrae il relativo frame posto
    % sul piano orizzontale restituendolo in output
    function imageFromMRI = extractyimage(mri, depth)
        if depth < -size(mri, 2) || depth > size(mri, 2)
            error('Depth is wrong');
        end
        %sqeeze rimuove le dimensioni unitarie quindi rende la
        %dimensione dell'immagine da MxYxN a MxN per Y=1
        imageFromMRI = squeeze(mri(:, depth, :));

        %ELENCO PIANI
        %frontal = squeeze(mri(:, :, depth));
        %sagittal = squeeze(mri(depth, :, :));
        %horizontal = squeeze(mri(:, depth, :));
    end

    % Visualizza le 3 viste ortogonali di un immagine 3D
    function show3dimage(img)
        figure;
        subplot(1,3,1);
        imshow(squeeze(img(:, :, round(size(img, 3)/2))), []);
        title('Piano mediano (X)');
        subplot(1,3,2);
        imshow(squeeze(img(:, round(size(img, 2)/2), :)), []);
        title('Piano mediano (Y)');
        subplot(1,3,3);
        imshow(squeeze(img(round(size(img, 1)/2), :, :)), []);
        title('Piano mediano (Z)');
    end


    % Mostra n immagini prese in input con i relativi nomi (opzionali)
    function imsshow(arrayImgs, namesImgs)
        numImgs = numel(arrayImgs);

        % Se namesImgs non viene fornito, lo inizializza con 'A'
        if ~exist('namesImgs', 'var') || isempty(namesImgs)
            namesImgs = zeros(0, 1);
        end

        numNamesImgs = numel(namesImgs);

        % Calcola il numero di righe e colonne per disporre le immagini nei subplot
        num_colonne = ceil(sqrt(numImgs));
        num_righe = ceil(numImgs / num_colonne);
```

```
            % Crea una nuova figura
            figure;

            % Itera su ciascuna immagine nell'array
            for i = 1 : numImgs
                % Crea un subplot e mostra l'immagine corrente
                subplot(num_righe, num_colonne, i);
                imshow(arrayImgs{i});

                % Imposta il titolo dell'immagine
                if i <= numNamesImgs
                    title(namesImgs{i});
                else
                    titleImage=['Img ' , num2str(i)];
                    title(titleImage);
                end
            end
        end


        % Converte un'immagine in scala di grigi (se necessario)
        function img = rgb2gray(img)
            if size(img, 3) == 3
                img = rgb2gray(img);
            end
        end

        % Calcola TP, TN, FP, FN in base ai pixel delle immagini
        function [TP, TN, FP, FN] = calculate_confusion_matrix(labels, testImage)
            % Calcola le matrici di confusione
            TP = sum(sum(labels & testImage));
            TN = sum(sum(~labels & ~testImage));
            FP = sum(sum(~labels & testImage));
            FN = sum(sum(labels & ~testImage));
        end
    end
end
```

2. Threshold Segmentation

Di seguito l'implementazione del codice per la segmentazione a soglia.

```
clear;
close all;

% Carica i percorsi delle immagini MRI del polmone
examplesFolders = Helpers.elenca_file_con_prefisso('Dataset', 'lung');

for j = 1 : size(examplesFolders,2)

    folderToSave = fullfile('Dataset', examplesFolders(j));
    inputImage  = imread(char(fullfile(folderToSave, 'trainingImage.png')));

    inputImage = Helpers.resize(inputImage, 256);

    % Erosione con elemento strutturale (diamante) con un raggio di 2
    structureElement = strel('diamond', 1);
    erodedImage = imerode(inputImage, structureElement);

    % Conversione immagine in scala di grigi se necessario
    erodedImage = Helpers.rgb2gray(erodedImage);

    %smussa gli angoli, porta a linee piu morbide nell'immagine. Visibili miglioramenti
    %sulla linea dei polmoni
    erodedImageMedfilt = medfilt2(erodedImage, [5, 5]);

    %Binarizzazione immagine con soglia graythresh Otsu
    binarizedImage = Helpers.otsubin(erodedImageMedfilt);
    imshow(binarizedImage);
    imageWithoutResults = bwareaopen(binarizedImage, 500);

    occurrences = binarizedImage - imageWithoutResults;
    imshow(occurrences);
    imwrite(logical(occurrences),char(fullfile(folderToSave,
'soglia_occorrenze_rilevate.png')));

    % Etichettatura delle regioni connesse nell'immagine binarizzata
    [label_matrix, num_labels] = bwlabel(occurrences);
    % Calcolo centroidi delle regioni connesse sulle occorrenze rilevate
    centroids = regionprops(label_matrix, 'Centroid');

    inputImageBin = Helpers.otsubin(inputImage);
    imshow(inputImage);

    hold on; % Abilita la sovrapposizione dei tracciati
    for i = 1:num_labels

        % Seleziona la regione bianca, sulle coordinate fornite,
        % dell'immagine iniziale binarizzata. In modo da ottenere area e
        % perimetro precisi
        region = bwselect(inputImageBin, centroids(i).Centroid(1),
centroids(i).Centroid(2));
        regionProps = regionprops(region, 'Area', 'Perimeter', 'Centroid');

        % Se le props delle regioni analizzate sono eccessive significa che
        % sono stati rilevati bordi errati (principalmente a causa di
        % tumori a contatto con le pareti del polmone)
        if ~isempty(regionProps) && (regionProps.Area > 600 || regionProps.Perimeter >
250)
            % Calcolo delle regionprops sulle occorrenze rilevate (con aree
            % e perimetri erosi, quindi minori)
            region = bwselect(occurrences, centroids(i).Centroid(1),
centroids(i).Centroid(2));
```

```
            regionProps = regionprops(region, 'Area', 'Perimeter', 'Centroid');
        end

        % Trova il perimetro della parte bianca dell'immagine binarizzata
        perimeterTrack = bwperim(region, 8);
        B = bwboundaries(perimeterTrack, 'noholes');
        for k = 1:length(B)
            boundary = B{k};
            plot(boundary(:, 2), boundary(:, 1), 'r', 'LineWidth', 1)
        end

        if ~isempty(regionProps)
            text(regionProps.Centroid(1), regionProps.Centroid(2), ...
                sprintf('Area: %d \n Perimetro: %.2f', regionProps.Area,
regionProps.Perimeter), 'Color', 'yellow', 'FontSize', 10);

            disp([' idxs: ' folderToSave  ' ' num2str(i) ' Area: '
num2str(regionProps.Area) ' Perimeter: ' num2str(regionProps.Perimeter)]);
        end
    end

    hold off;% Disbilita la sovrapposizione dei tracciati

    exportgraphics(gcf, char(fullfile(folderToSave, 'segmentazione_a_soglia.png')));

end
```

Per calcolare l'area e il perimetro in modo più preciso nella segmentazione a soglia si è scelto di procedere nel seguente modo:

- Calcolo dei centroidi delle regioni contenute nelle **occorrenze rilevate** (ottenute da manipolazioni precedenti derivanti dalla segmentazione a soglia).
- Estrapolo le regioni a partire dall'**input image,** opportunamente binarizzata, e dai centroidi precedentemente calcolati (che essendo stata sottoposta a minori manipolazioni conserverà meglio i dati su area e perimetro).
- Calcolo le *regionprops* dall'input image, se i dati di Area e Perimetro non sono superiori ad una certa soglia verranno considerati corretti, altrimenti verranno calcolate le regionprops di occorrenze rilevate.

Con l'utilizzo di input image possono essere erroneamente considerate aree ben più estese a causa di tumori a contatto con la parete del polmone. A questo scopo è stata introdotta la soglia massima `(regionProps.Area > 600 || regionProps.Perimeter > 250)`.

3. Marker-Controlled Segmentation

Di seguito l'implementazione del codice per la segmentazione marker controlled [2][3].

```
clear;
close all;

% Carica i percorsi delle immagini MRI del polmone
examplesFolders = Helpers.elenca_file_con_prefisso('Dataset', 'lung');


for j = 1 : size(examplesFolders,2)

    folderToSave = fullfile('Dataset', examplesFolders(j));
    inputImage  = imread(char(fullfile(folderToSave, 'trainingImage.png')));
    inputImage = Helpers.rgb2gray(inputImage);

    % Ridimensionamento dell'immagine
    inputImage = Helpers.resize(inputImage, 256);

  % Opening-by-Reconstruction: eseguo imerode ed imreconstruct.

    % Escludo porzioni bianche troppo piccole per essere riconducibili a
    % tumori (in genere si tratta di alveoli polmonari)
    se = strel("diamond", 3);
    Ie = imerode(inputImage,se);
    i_open_by_reconstr = imreconstruct(Ie, inputImage);
  %____________________


  % Opening-Closing-by-Reconstruction
    i_open_by_reconstr_dilated = imdilate(i_open_by_reconstr, se);

    %imreconstruct(marker, mask). Il marker è l'immagine erosa che
    %ricostruiamo con mask.
    %Avendo dilatato Iobrd marker potrebbe essere "maggiore" di mask quindi
    %uso imcomplement

    %imcomplement: complement of image 255 - pixel_intensity
    Iobrcbr = imreconstruct(imcomplement(i_open_by_reconstr_dilated), imcomple-
ment(i_open_by_reconstr));
    Iobrcbr = imcomplement(Iobrcbr);
  %____________________

  %CALCOLO GRADIENTE

    %[gmag_alternativo, gdir_alternativo] = imgradient(I);

    % Matrici di Sobel
    sobel_x = [-1 0 1; -2 0 2; -1 0 1]; % Sobel kernel per Gx
    sobel_y = [-1 -2 -1; 0 0 0; 1 2 1]; % Sobel kernel per Gy

    % Applicazione della convoluzione dell'immagine con i kernel Sobel
    gradiente_x = imfilter(double(Iobrcbr), sobel_x);
    gradiente_y = imfilter(double(Iobrcbr), sobel_y);

    % Calcolo del gradiente con Sobel
    gradient_magnitude = sqrt(gradiente_x.^2 + gradiente_y.^2);

    gradient_direction = atan2(gradiente_y, gradiente_x);
    gradient_direction_deg = rad2deg(gradient_direction);

  %____________________

    %Regional Maxima of Opening-Closing by Reconstruction
```

```
    fgm = imregionalmax(Iobrcbr);

    %Modified Regional Maxima Superimposed on Original Image
    se2 = strel(ones(3, 3));
    %L'operazione di chiusura morfologica è una dilatazione seguita da un'erosione,
utilizzando lo stesso elemento strutturante per entrambe le operazioni.
    fgm2 = imclose(fgm, se2);

    fgm3 = bwareaopen(fgm2, 20);
    I3 = labeloverlay(inputImage, fgm3);

    bw  = imbinarize(Iobrcbr);

    distance_transform = bwdist(bw);
    distance_labels = watershed(distance_transform);

    % La distance_transform rende il polmone bianco
    % quindi la linea di cresta si posizionerà sul massimo (bianco).
    % Watershed Ridge Lines.
    bgm = distance_labels == 0;

    maschera = bgm | fgm3;
    gmag2 = imimposemin(gradient_magnitude, maschera);
    L = watershed(gmag2);

    %Nella labels: L == 0 dilatate con elemento strutturale. Valori bgm a 2. Valori
fgm3 a 3.
    labels = imdilate(L==0, ones(3,3)) + 2 * bgm + 3 * fgm3;

    %Markers and Object Boundaries Superimposed on Original Image
    I4 = labeloverlay(inputImage, labels);
    imshow(I4);

    exportgraphics(gcf, char(fullfile(folderToSave, 'labels_marker_controlled_.png')));

    % Colora il tumore (etichette assegnate in base ai marcatori)
    colored_img = label2rgb(L);
    bgm_filled = imfill(bgm, 'holes');

    bgm_filled_AND_fgm3 = bgm_filled & fgm3;

    imshow(inputImage)
    hold on


    [label_matrix, num_labels]  = bwlabel(bgm_filled_AND_fgm3);
    region_props_label_matrix = regionprops(label_matrix, 'Centroid', 'Area',
'Perimeter');

    image_size = size(inputImage);
    occorrenze_rilevate = false(image_size(1), image_size(2));

    if ~isempty(region_props_label_matrix)

        for i = 1:num_labels
            region = bwselect(L, region_props_label_matrix(i).Centroid(1),
region_props_label_matrix(i).Centroid(2));
            regionProps = regionprops(region, 'Area', 'Perimeter', 'Centroid');

            % Se il centroide della rilevazione non è nel tumore per evitare
            % regioni errate del bwselect vengono impostate soglie di
```

```
            % coerenza. Se area o perimetro sono maggiori di queste soglie
            % si utilizzerà la regione processata (con area e perimetro
            % meno precisi)
            if regionProps.Area > 600 || regionProps.Perimeter > 250
                regionProps = region_props_label_matrix(i);
                region = label_matrix == i;
            end

            if ~isempty(regionProps)
                text(regionProps.Centroid(1), regionProps.Centroid(2), sprintf('Area:
%d \n Perimetro: %.2f', regionProps.Area, regionProps.Perimeter), 'Color',
'yellow','FontSize', 6);
            end

            %aggiorno la matrice logica di occorrenze rilevate basandomi
            %sulla selezione della regione iniziale trovata tramite il
            %centroide
            occorrenze_rilevate = occorrenze_rilevate | region;
        end
    end

    % Converti la matrice logica in uint8
    uint8Overlay = uint8(occorrenze_rilevate);

    % Converti l'immagine uint8 in una immagine RGB
    rgbOverlay = cat(3, uint8Overlay*255, uint8Overlay*0, uint8Overlay*0);  % Rosso
    overlaySeg = imshow(rgbOverlay);
    overlaySeg.AlphaData = 0.5;
    hold off;
    exportgraphics(gcf, char(fullfile(folderToSave,
'segmentazione_marker_controlled.png')));

    imshow(occorrenze_rilevate, []);
    imwrite(logical(occorrenze_rilevate), char(fullfile(folderToSave,
'marker_controlled_occorrenze_rilevate.png')));
end
```

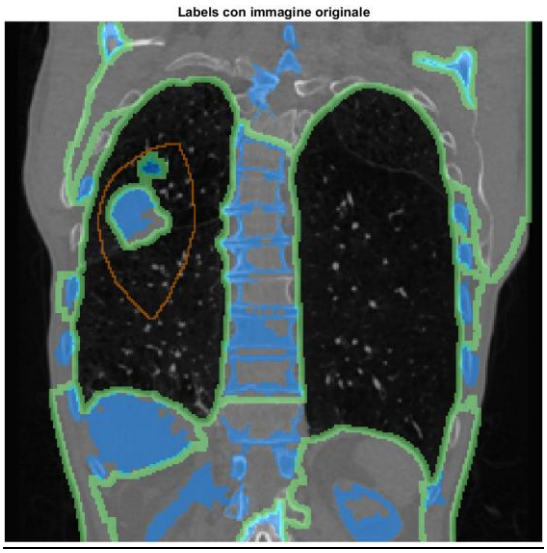

Figura 7 Immagine di input con marcatori sovrapposti

6. Dataset & Performance

I due approcci proposti sono stati testati su una collezione di 63 file di risonanza magnetica (.nii.gz), da cui sono state estrapolate le immagini da processare e i riscontri di verità. Per valutare l'efficacia degli algoritmi è stata utilizzata una matrice di confusione in cui si definiscono:

- ***TP - true positive***: pixel in cui è correttamente identificato un tumore;
- ***TN - true negative***: pixel in cui correttamente non è identificato alcun tumore.
- ***FP - false positive***: pixel in cui è stato erroneamente identificato un tumore.
- ***FN - false negative***: pixel classificati erroneamente come non appartenenti ad un'area tumorale.

In base alla matrice di confusione sono state ricavate le seguenti metriche di valutazione:

Acc (Accuracy): rappresenta il numero totale di pixel correttamente segmentati sul numero totale di pixel.

$$\frac{TP+TN}{TP+TN+FP+FN}$$

TPR (True Positive Rate): Indica la proporzione di pixel positivi (nel nostro caso, che identificano un'area tumorale) correttamente identificati dal modello rispetto al totale dei pixel positivi reali presenti nel dataset.

$$\mathrm{TPR} = \frac{TP}{TP + FN}$$

TNR (True Negative Rate): Indica la proporzione di pixel negativi (nel nostro caso, che non identificano un'area tumorale) correttamente identificati dal modello rispetto al totale dei pixel in cui non è presente alcun tumore.

$$TNR = \frac{TN}{TN+FP}$$

Nel seguito vengono riportati i dati delle performance generati in modo automatizzato dopo la modifica manuale di alcuni parametri.

4. Performance Threshold Segmentation

```
strel("diamond", 2);
medfilt2(erodedImage, [5, 5]);
bwareaopen(binarizedImage, 500);
```

Threshold Segmentation	
TP	2306
TN	4103472
FP	3610
FN	19380

ACC = 0.9944

TPR = 0.1063

TNR = 0.9991

```
strel("diamond", 1);
medfilt2(erodedImage, [5, 5]);
bwareaopen(binarizedImage, 500);
```

Threshold Segmentation	
TP	3303
TN	4102589
FP	4493
FN	18383

ACC = 0.9945

TPR = 0.1523

TNR = 0.9989

```
removed imerode
medfilt2(erodedImage, [5, 5]);
bwareaopen(binarizedImage, 500);
```

Threshold Segmentation	
TP	2778
TN	4097037
FP	10045
FN	18908

ACC = 0.9930

TPR = 0.1281

TNR = 0.9976

```
strel("diamond", 1);
medfilt2(erodedImage, [2, 2]);
bwareaopen(binarizedImage, 500);
```

Threshold Segmentation	
TP	3131
TN	4100371
FP	6711
FN	18555

ACC = 0.9939

TPR = 0.1444

TNR = 0.9984

```
strel("diamond", 1);
medfilt2(erodedImage, [7, 7]);
bwareaopen(binarizedImage, 500);
```

Threshold Segmentation	
TP	2277
TN	4103774
FP	3308
FN	19409

-----THRESHOLD-----

ACC = 0.9945

TPR = 0.1050

TNR = 0.9992

```
strel("diamond", 1);
medfilt2(erodedImage, [5, 5]);
bwareaopen(binarizedImage, 400);
```

Threshold Segmentation	
TP	2827
TN	4103076
FP	4006
FN	18859

ACC = 0.9945

TPR = 0.1304

TNR = 0.9990

```
strel("diamond", 1);
medfilt2(erodedImage, [5, 5]);
bwareaopen(binarizedImage, 700);
```

Threshold Segmentation	
TP	3303
TN	4102589
FP	4493
FN	18383

ACC = 0.9945

TPR = 0.1523

TNR = 0.9989

5. Performance Marker Controlled Segmentation

Nel seguito vengono riportati i dati delle performance generati in modo automatizzato dopo la modifica manuale di alcuni parametri.

```
strel("diamond", 3);
strel(ones(3, 3));
bwareaopen(fgm2, 20);
```

Marker Controlled Segmentation	
TP	2707
TN	4104577
FP	2505
FN	18979

ACC = 0.9948

TPR = 0.1248

TNR = 0.9994

```
strel("diamond", 2);
strel(ones(3, 3));
bwareaopen(fgm2, 20);
```

Marker Controlled Segmentation	
TP	2923
TN	4099503
FP	7579
FN	18763

ACC = 0.9936

TPR = 0.1348

TNR = 0.9982

```
strel("diamond", 1);
```

```
strel(ones(3, 3));
bwareaopen(fgm2, 20);
```

Marker Controlled Segmentation	
TP	2917
TN	4099209
FP	7873
FN	18769

ACC = 0.9935

TPR = 0.1345

TNR = 0.9981

```
strel("diamond", 4);
strel(ones(3, 3));
bwareaopen(fgm2, 20);
```

Marker Controlled Segmentation	
TP	2260
TN	4106467
FP	615
FN	19426

ACC = 0.9951

TPR = 0.1042

TNR = 0.9999

```
strel("diamond", 3);
strel(ones(2, 2));
bwareaopen(fgm2, 20);
```

Marker Controlled Segmentation	
TP	2698
TN	4104621
FP	2461
FN	18988

ACC = 0.9948

TPR = 0.1244

TNR = 0.9994

```
strel("diamond", 3);
strel(ones(4, 4));
bwareaopen(fgm2, 20);
```

Marker Controlled Segmentation	
TP	2707
TN	4104518
FP	2564
FN	18979

ACC = 0.9948

TPR = 0.1248

TNR = 0.9994

```
strel("diamond", 3);
strel(ones(3, 3));
bwareaopen(fgm2, 5);
```

Marker Controlled Segmentation	
TP	2707
TN	4104577
FP	2505
FN	18979

ACC = 0.9948

TPR = 0.1248

TNR = 0.9994

```
strel("diamond", 3);
strel(ones(3, 3));
bwareaopen(fgm2, 40);
```

Marker Controlled Segmentation	
TP	2650
TN	4104577
FP	2505
FN	19036

ACC = 0.9948

TPR = 0.1222

TNR = 0.9994

7. Conclusione

Il risultato migliore tra tutti è stato raggiunto tramite la **segmentazione a soglia** impostando i parametri dell'erosione con elemento strutturale diamante a 1 (per ripulire l'immagine ed escludere gli alveoli), filtro mediano a [5, 5] (per smussare le linee) e bwareaopen con 500 pixel come parametro (per l'esclusione del tumore prima della sottrazione).

Il risultato migliore per la **segmentazione marker controlled** è stato ottenuto impostando i parametri di erosione con elemento strutturale diamante a 3 (per ripulire l'immagine ed escludere gli alveoli), erosione con elemento strutturale matrice di uno (per riempire piccoli buchi) e bwareaopen con parametro 20 (per escludere piccole imperfezioni). Impostando l'elemento strutturale diamante a due si otterrebbe un miglior TPR ma un numero considerevolmente maggiore di falsi positivi.

In linea generale quindi la segmentazione a soglia si è comportata meglio, confrontando le due sulle immagini risultati si è potuto notare che la segmentazione a soglia è migliore per quanto riguarda i tumori di piccole dimensioni presenti all'interno dei polmoni (lontani quindi dalle pareti polmonari), al contrario, la segmentazione marker controlled si è dimostrata leggermente migliore nel riconoscere i tumori aderenti alle superfici laterali (che comunque rimangono un grosso problema, di fatti, come detto in precedenza ha prestazioni generalmente peggiori nei nostri test).

Osservazione: le MRI, contenendo meno dettaglio rispetto alle CT Scan (Computed Tomography Scan), possono aver penalizzato molto le performance ("Computed tomography (CT) scan images are preferred to be used in this research, due to low noise and better clarity compared to X-Ray and MRI images" [6]).

Il link al repository contenente il codice del progetto è in Bibliografia [8].

8. Bibliografia

[1] https://it.wikipedia.org/wiki/Segmentazione_di_immagini

[2] https://it.mathworks.com/help/images/ref/watershed.html

[3] https://it.mathworks.com/help/images/marker-controlled-watershed-segmentation.html

[4] https://www.ibm.com/topics/image-segmentation#Traditional+image+segmentation+techniques

[5] https://download.slicer.org/

[6]https://www.sciencedirect.com/science/article/pii/S187705091732433X/pdf?md5=c110faa01ba19e1d92afe1342080fc10&pid=1-s2.0-S187705091732433X-main.pdf

[7]https://citeseerx.ist.psu.edu/document?repid=rep1&type=pdf&doi=d50e8974d9e80ee3c0446dfe18707a07eaef067e

[8] https://github.com/hakunamatata95/ImageProcessing/settings