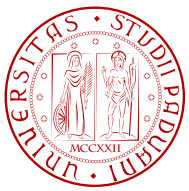
Annaclaudia Montanino

Segmentazione di immagini istologiche basata su grafi

Tesi di laurea triennale



Relatore: prof. Enrico Grisan
Correlatrice: dott. Elisa Veronese

Università degli Studi di Padova
Facoltà di Ingegneria
Dipartimento di Ingegneria dell'Informazione
Settembre 2012

Indice

Elenco delle figure

Sommario

Lo scopo di questa tesi è mostrare una possibile implementazione del metodo di diagnosi automatica (basata su grafi) di adenocarcinoma a partire da immagini istologiche di tessuti esofagei. L'approccio scelto è di tipo strutturale: il tessuto viene completamente caratterizzato dalle relazioni spaziali esistenti tra i propri componenti cellulari e tali relazioni sono rappresentabili esplicitamente con un grafo colorato. E' possibile perciò identificare un'immagine istopatologica con il suo grafo corrispondente e da quest'ultimo ricavare i parametri necessari alla diagnosi. Dopo una breve introduzione in cui viene illustrato lo stato dell'arte di tale materia, segue nel Capitolo 1 (1) la definizione sintetica di alcune nozioni mediche quali *esofago*, *giunzione esofago-gastrica*, *esofago di Barrett*, *Adenocarcinoma esofageo/cardiale* utili alla comprensione del programma successivamente descritto. Nel Capitolo 2 (2) vengono illustrati in dettaglio i passaggi che consentono di associare l'immagine istopatologica al grafo colorato e nel Capitolo 3 (3) vengono definite le features (parametri estrapolabili dal grafo) sulla cui valutazione si basa la diagnosi automatica. Nelle conclusioni (4) sono infine raccolti i risultati derivanti dall'applicazione del programma ad alcune immagini istologiche di tessuti esofagei.

Introduzione

Nella medicina moderna le immagini sono uno dei principali strumenti per quanto riguarda la diagnosi e la stadiazione di patologie come ad esempio il cancro. Ad oggi, tuttavia, è ancora l'uomo, nella figura dell'anatomopatologo, che, osservando immagini tratte da prelievi bioptici, prende decisioni sulla presenza o meno di regioni cancerose e sul loro grado di malignità. E' evidente però che tale analisi, basandosi su un'interpretazione visiva, è soggetta ad una considerevole variabilità inter e intra-osservatore. Per ridurre questa variabilità, numerosi studi stanno evidenziando la possibilità di effettuare, a partire dall'immagine, una diagnosi automatica sfruttando metodi computazionali basati sulle proprietà dell'immagine stessa. Per quantificare l'immagine di un tessuto sono stati usati finora diversi approci.

- Un tipo di approccio è quello **morfologico** in cui si valutano la forma e le dimensioni dei componenti cellulari in una data immagine. In **[3]** è stato sfruttato il metodo dei contorni attivi allo scopo di ottenere l'esatta forma e dimensione dei nuclei dell'immagine. Da queste informazioni sono quindi state ricavate dieci features (per la definizione di *feature* si veda 3.1) (area, perimetro, simmetria, rotondità, etc.)per ogni nucleo. Tuttavia, essendo le immagini istopatologiche di natura molto complessa, per seguire un procedimento di questo tipo è necessario un grande sforzo a livello di estrazione di contorni.

- Un approccio che invece non necessita di una fase iniziale di segmentazione è quello basato sull'**intensità dei pixels** dell'immagine. In **[4]**, ad esempio, dopo aver diviso l'immagine in 256 elementi quadrati (32×32 pixels), a partire dall'istogramma di ognuno di questi elementi quadrati, vengono calcolate 15 features (media, deviazione standard, simmetria, convessità, etc.). Se però questo approccio viene utilizzato su immagini di prelievi istologici colorati con ematossilina e eosina la differenza fra le varie delle distribuzioni di intensità non è apprezzabile.

- Un altro approccio è quello basato sulla caratterizzazione della **texture** di un tessuto. Per ottenere tale caratterizzazione si ricavano delle *textural features* - come: secondo momento angolare, contrasto, la correlazione, varianza, entropia **[5]**, omogeneità, dissimilarità **[6]**, etc. - a partire da matrici di co-occorrenza (per la definizione di *matrice di co-occorrenza* si veda p.414 di **[8]**).

- Infine con un approccio di tipo **strutturale** -approccio che si è poi adottato in questa tesi- il tessuto viene caratterizzato dalla distribuzione spaziale

dei propri componenti cellulari. Le relazioni fra le diverse componenti sono rappresentate con un grafo (2.1.1) come in [**1**] e in [**5**] e su questo vengono calcolate le features.

Lo scopo di questa tesi è illustrare in dettaglio una delle possibili implementazioni del procedimento descritto in [**1**] per la diagnosi automatica di adenocarcinoma del colon e verificare che questo algoritmo sia valido anche per la diagnosi su tessuti esofagei.

Capitolo 1

Esofago e Adenocarcinoma

L'obiettivo di questo capitolo è introdurre in modo efficace, seppur non approfondito, la patologia denominata *Adenocarcinoma*. A tale scopo si propone inizialmente una breve descrizione dell'anatomia macroscopica e microscopica dell'esofago, mentre, a seguire, si descrive la patologia sopracitata.

1.1 L'Esofago

L'esofago è un organo dell'apparato digerente che, decorrendo pressoché rettilineo per circa 25 cm, congiunge la faringe allo stomaco tramite un orifizio detto cardias. Le sue funzioni principali sono: consentire il passaggio del bolo alimentare e impedire la risalita del contenuto gastrico nella cavità orale. Rilevante è inoltre l'attività di lubrificazione delle pareti interne che facilita la discesa del cibo. Dal punto di vista microscopico l'esofago presenta, similmente al resto dell'intero tubo gastrointestinale, una parete costituita di quattro principali strati. Procedendo dal lume dell'organo verso la superficie esterna si succedono le seguenti lamine:

- la ***mucosa*** è la membrana più interna e in essa si possono distinguere: l'epitelio di tipo pavimentoso stratificato non cheratinizzato che protegge la mucosa da eventuali materiali contundenti ingeriti; la tonaca propria, costituita da un tessuto connettivo areolare e/o linfatico che supporta l'epitelio e lo connette alla muscolaris mucosae; e infine la muscolaris mucosae, costituita da due sottili strati di fibre muscolari lisce e tessuto elastico che consente movimenti localizzati della mucosa.

- la ***sottomucosa*** connette la mucosa alla muscolaris externa ed è costituita da tessuto connettivo lasso, flessibile ed elastico grazie al quale forma le pieghe della mucosa. In questa lamina sono inoltre presenti le ghiandole esofagee: piccole ghiandole ramificate, acinose di tipo mucoso distribuite in prevalenza lungo i solchi che la mucosa presenta in superficie. Il loro dotto escretore si apre a livello della superficie della mucosa dopo aver attraversato, nel suo passaggio attraverso la tonaca propria, un nodulo linfatico.

- la ***muscolaris externa***, costituita da due strati di muscolatura liscia (uno interno circolare ed uno esterno longitudinale), contribuisce alla spinta del bolo con le sue contrazioni.

- la ***tonaca avventizia*** avvolge esternamente l'esofago e lo collega agli organi vicini.

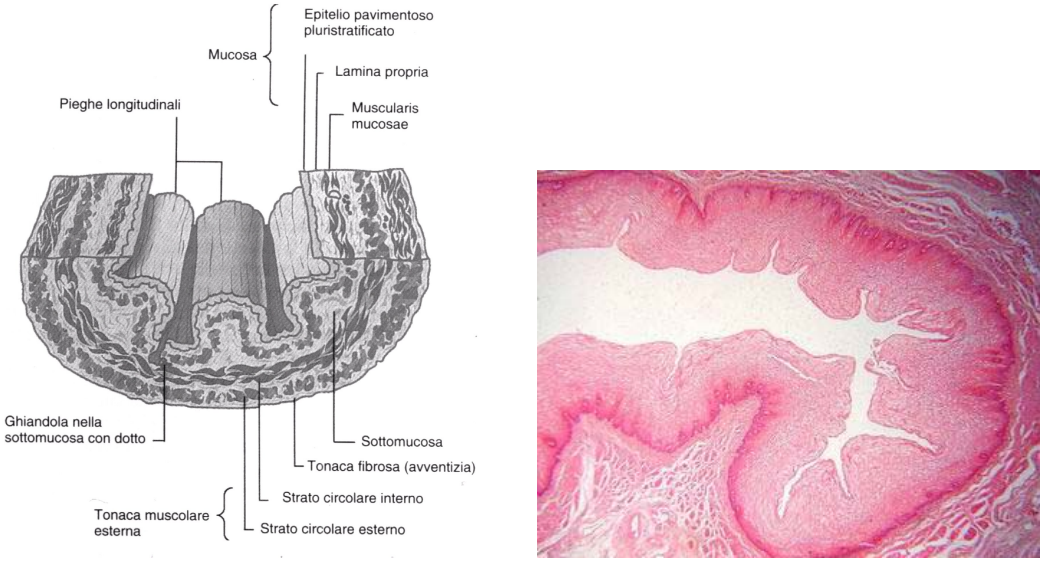


Figura 1.1: Tonache esofagee

1.2 Giunzione esofago-gastrica

La descrizione precedente si riferisce all'anatomia esofagea di soggetti adulti normali, tuttavia durante la vita fetale l'epitelio che riveste esofago e stomaco è composto da cellule cilindriche ed elementi ghiandolari che secernono muco. E' con lo sviluppo che l'esofago viene rivestito, definitivamente, da epitelio pavimentoso stratificato, ad eccezione di alcune isole di mucosa a rivestimento epiteliale cilindrico che possono permanere fino all'età adulta. La **giunzione anatomica (EsophagoGastric Junction, EGJ)** tra esofago e stomaco si definisce cardias, tuttavia non è detto che questa coincida con la **giunzione istologica (SquamoColumnar Junction, SCJ)** (limite tra epitelio pavimentoso stratificato ed epitelio cilindrico). Come evidenziato in [**8**] la mucosa di questo tratto, mucosa cardiale, essendo un ibrido tra questi due epiteli (presenta la struttura ghiandolare caratteristica della mucosa gastrica pilorica) sfugge alle classificazioni e viene considerata da alcuni un residuo del rivestimento embrionale dell'intestino anteriore, da altri, se dotata di alcune caratteristiche che specificheremo in seguito, viene ritenuta una condizione prepatologica che definiremo nel prossimo paragrafo: l'esofago di Barrett.

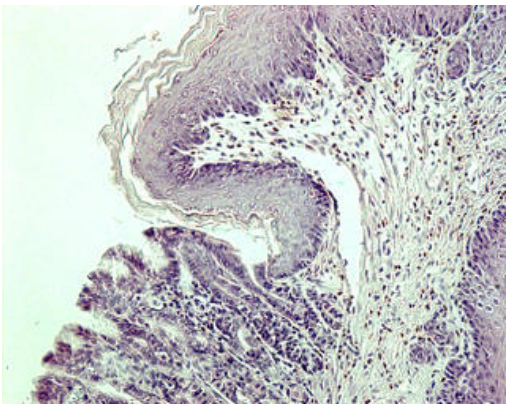


Figura 1.2: Giunzione esofago-gastrica

1.3 Esofago di Barrett

L'**Esofago di Barrett (Barrett's Esophagus, BE)** viene attualmente definito come presenza, rilevabile endoscopicamente, di epitelio metaplastico intestinale nel tratto distale dell'esofago per un'estensione $> 3cm$, oppure $< 3cm$ nel caso sia presente epitelio cilindrico specializzato. Lo sviluppo a tale livello di metaplasia (sostituzione di un tipo cellulare differenziato con un altro tipo cellulare differenziato) intestinale, la sua dimostrazione esclusivamente in sede cardiale e l'instabilità genomica che lo contraddistingue, costituisce il serbatoio naturale allo sviluppo dell'adenocarcinoma esofago-cardiale (in drammatico aumento negli ultimi venti anni).

1.4 Adenocarcinoma esofageo/cardiale

L'adenocarcinoma esofageo interessa solitamente il tratto inferiore dell'esofago ma, più raramente, può essere rinvenuto anche nell'esofago medio o superiore. E' una neoplasia che trae origine da epitelio colonnare metaplasico, oppure da mucosa gastrica eterotopica o da ghiandole sottomucose. Tale neoplasia è solitamente preceduta da un'alterata crescita tissutale: avvengono in particolare alterazioni quali **ingrossamento dei nuclei**, **pluristratificazione dell'epitelio colonnare**, **diminuzione della quantità dello stroma**, e in generale **vengono meno le strutture ghiandolari**. Gli elementi neoplastici posso presentare una scarsa coesività ed infiltrano i tessuti in modo diffuso. La diagnosi di questa patologia viene effettuata sulla base di esami che consentono di esaminare lo stato dell'esofago: radiografia del torace, esofagogramma oppure esofagoscopia. Durante l'esofagoscopia è inoltre possibile, qualora necessario, prelevare un campione di tessuto (biopsia) che sarà esaminato al microscopio per accertare la presenza di neoplasie. Quello che si descriverà nei prossimi capitoli è un metodo di diagnosi tumorale e stadiazione automatica che ha lo scopo di aiutare la figura dell'anatomopatologo nella sua valutazione.

Capitolo 2

Metodi

In questa sezione si vuole descrivere il procedimento seguito per passare dall'immagine istologica al corrispondente grafo colorato sul quale si baserà il calcolo delle features che definiremo nel prossimo capitolo. Il procedimento che è stato implementato segue le linee guida proposte in [**1**]. Nei successivi paragrafi vengono fornite la descrizione e l'implementazione di tale metodo.

2.1 Nozioni base

Si dà innanzitutto la definizione di alcuni concetti base per la comprensione del procedimento seguito che sarà illustrato nei paragrafi successivi.

2.1.1 Grafo

Si definisce Grafo G una coppia (N, A) dove N è un insieme di nodi (o vertici) ed A è una collezione di coppie (u, v) dette archi con u e v vertici terminali appartenenti a N. Un grafo consente di rappresentare le relazioni esistenti tra coppie di oggetti appartenenti all'insieme N.

Un grafo si dice **orientato** se tutte le coppie (u, v) sono coppie ordinate, viceversa si dice **non orientato**.

Due nodi u e v si dicono **adiacenti** se esiste un arco a i cui vertici terminali sono u e v, mentre u e v si dicono **connessi** se esiste una sequenza di archi a_1, a_2, $\ldots$, a_n tale che $a_1 = (u, x)$, $a_2 = (x, y)$, ..., $a_n = (z, v)$

Dato un grafo G non pesato (cioè in cui tutti gli archi hanno peso unitario), la **lunghezza di un percorso** dal nodo u al nodo v si definisce come il numero di archi appartenenti alla sequenza $a1, a2, \ldots, an$ che connette u a v.

Si definisce **percorso minimo** (o distanza) da un vertice u ad un vertice v la lunghezza del percorso di lunghezza minima tra u e v; se tale percorso non esistesse la distanza fra u e v si stabilisce essere infinita.

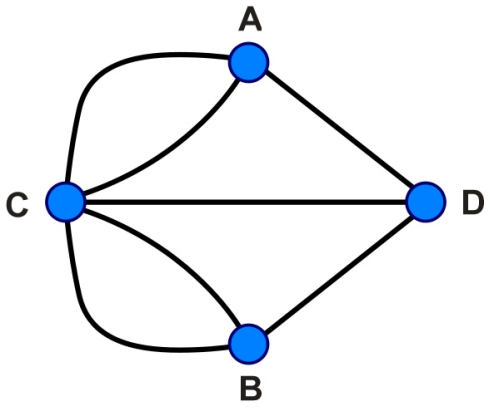


Figura 2.1: Grafo generico

2.1.2 Struttura di una ghiandola

Rispetto agli studi precedenti, l'idea nuova dell'articolo [**1**] è considerare portatrice di informazione non solo la posizione nello spazio dei nuclei delle cellule cilindriche come in [**5**], ma anche la posizione di altre componenti, quella luminale e quella stromale che verranno di seguito definite. In una ghiandola le due componenti principali sono l'epitelio ghiandolare (secernente) e lo stroma che ha invece funzioni di sostegno. Le cellule cilindriche dell'epitelio sono disposte circolarmente intorno ad un lume a formare una ghiandola che si collega, grazie al connettivo stromale, ad altre ghiandole. Poiché l'adenocarcinoma, come precedentemente spiegato, va ad alterare questa struttura ben definita, è possibile stabilire la cancerosità di un tessuto e il suo grado di cancerosità mediante la **quantificazione** di questo **disordine**. A tale scopo l'informazione aggiuntiva contenuta nella disposizione di due ulteriori componenti (quella stromale e quella luminale) è sicuramente rilevante.

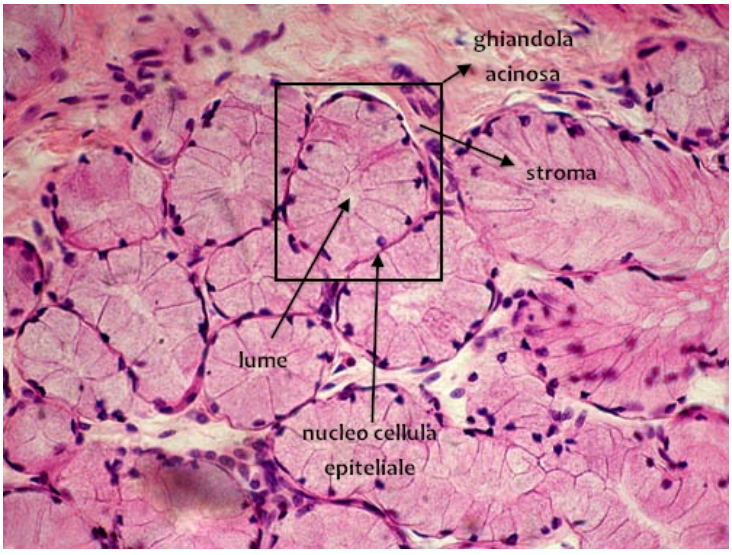


Figura 2.2: Ghiandola acinosa di tipo mucoso

2.2 Costruzione del grafo colorato

Il procedimento seguito per associare a ciascuna immagine un grafo colorato è articolabile nei sottopassaggi che ci si appresta qui ad approfondire.

2.2.1 Dall'immagine RGB all'immagine quantizzata

Osservando ad occhio nudo un'immagine istopatologica colorata con ematossilina eosina è facile apprezzare che i colori presenti sono principalmente tre: viola, rosa e bianco, colori che nel nostro caso è possibile associare rispettivamente alla componente nucleare, a quella stromale e a quella luminale. Considerando che l'obiettivo finale è ottenere un grafo che rispecchi i rapporti strutturali fra le tre componenti (clusters), è utile ricavare dall'immagine RGB un'immagine il cui numero di diversi colori visibili sia esattamente tre (viola, rosa e azzurro, in sostituzione del bianco). Per attuare questo processo di quantizzazione del colore si deve associare ciascun pixel dell'immagine RBG ad uno dei tre clusters.

Si è scelto innanzitutto di convertire i valori RGB dei pixel dell'immagine nei corrispondenti valori dello spazio L*a*b* (come suggerito in [**7**]) poiché nello spazio L*a*b* la distanza euclidea fra i colori riflette meglio la differenza nei colori percepiti. Si è poi eliminata la componente L* (luminosità) poiché l'informazione sul colore è contenuta nelle sole componenti a* (intervallo verde-rosso) e b* (intervallo blu-giallo). I pixel risultano quindi essere degli oggetti il cui valore è la coppia (a, b) invece che la tripletta (r, g, b).

Con l'algoritmo di clustering *kmeans*, che si fonda sul criterio di minimizzazione della varianza intracluster, è stato possibile partizionare tutti gli elementi (pixel) dell'insieme immagine in k gruppi (nel nostro caso $k = 3$). La funzione *kmeans* di MATLAB che è stata utilizzata implementa questo algoritmo (di complessità $0(N * i * q)$ con N numero di pixel, i numero di cluster, q numero di iterazioni per convergere) ha come output un vettore di indici IDX: il primo pixel dell'immagine appartiene al cluster indicato da $IDX(1)$, il secondo a quello indicato da $IDX(2)$ e così via.

E' stata successivamente calcolata per ciascun cluster la media delle intensità dei pixel appartenenti al dato cluster. Da questa informazione è infatti possibile stabilire quale dei tre indici assegnati dal kmeans corrisponda a ciascun cluster: l'indice corrispondente alla componente nucleare (viola) sarà quello del cluster con intensità media minore dei tre, così come l'indice corrispondente alla componente luminale (bianco) sarà quello del cluster con intensità media massima fra le tre e infine l'indice corrispondente alla componente stromale (rosa) sarà quello del cluster avente intensità media intermedia.

E' stata dunque usata una mappa ad hoc per passare dalla matrice di indici all'immagine quantizzata a livelli di grigio (nero in sostituzione del viola, grigio in sostituzione del rosa e bianco) con lo scopo di poter utilizzare su questa degli operatori morfologici per ridurre l'eventuale rumore e gli errori di clustering. La

mappa costruita è una matrice di dimensioni $K \times 3$ ($K = 3$ nel nostro caso) in cui sulla riga *i-esima* $(i, :)$ si ha la tripletta (r, g, b) del colore corrispondente al cluster i. Dopo aver filtrato l'immagine a livelli di grigio si è ottenuta l'immagine quantizzata colorata sostituendo ai pixel di colore nero, grigio, bianco rispettivamente pixel di colore viola, rosa e azzurro.

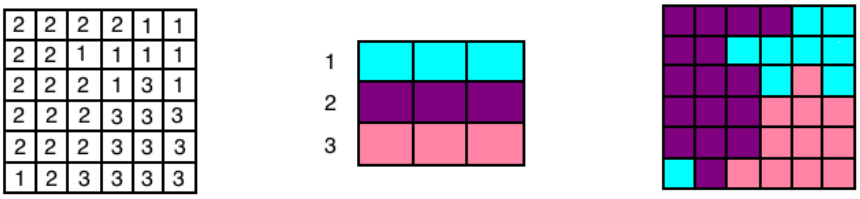


Figura 2.3: Mappa

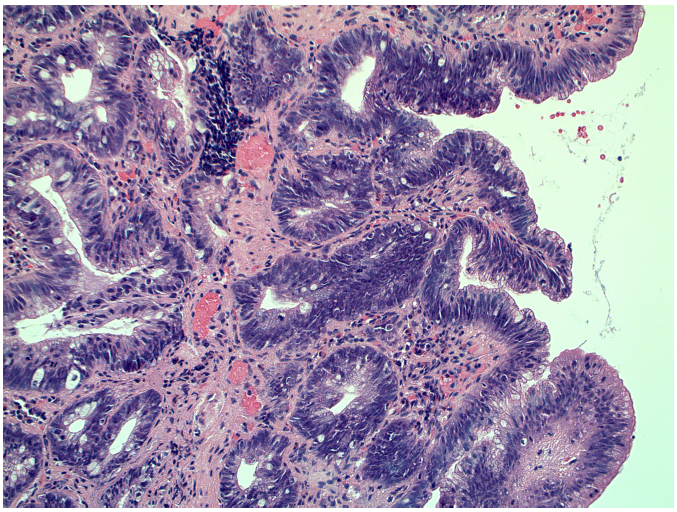


Figura 2.4: Immagine RGB

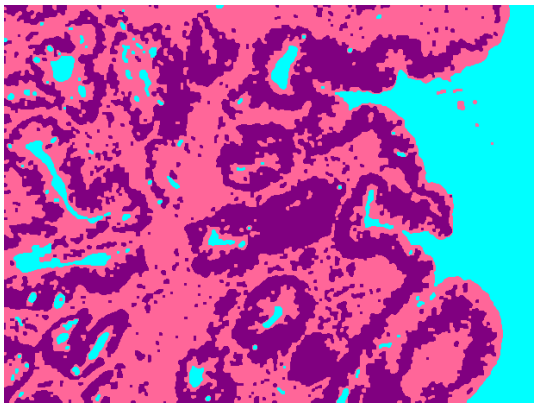
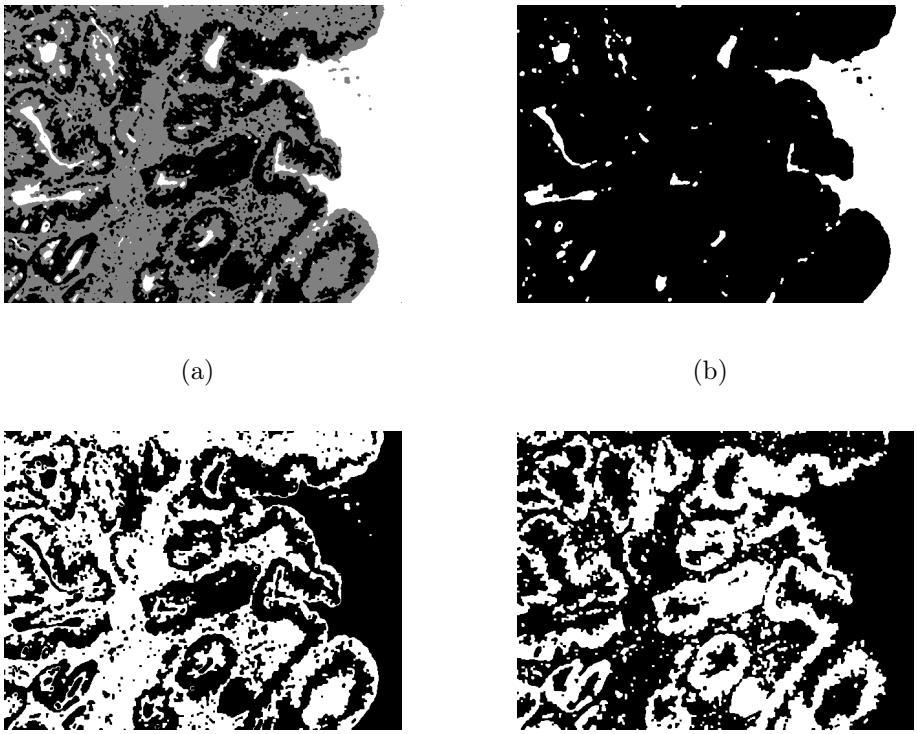


Figura 2.5: Immagine quantizzata

2.2.2 Dall'immagine quantizzata all'immagine di primitive

L'obiettivo di questo passo è ottenere un insieme di primitive circolari (cerchi) che rappresentino l'immagine e che siano quindi dei validi nodi per il grafo che si vuole ottenere. Si è scelto di non utilizzare l'algoritmo iterativo proposto in **[2]** con la speranza di riuscire a snellire un programma già molto pesante. E' stata perciò creata la funzione (vedi A.2) che a partire da un immagine binaria fornisce in output i centri e le aree dei cerchi che rappresentano al meglio le regioni in cui la matrice assume valore unitario. L'idea è quella di assegnare ogni pixel $P = 1$ dell'immagine binaria alla circonferenza di raggio maggiore che contiene, oltre a P stesso, solo pixel non nulli. Segue un esempio che definisce il metodo di allocazione delle primitive del cluster1.



(a) (b)

(c) (d)

Figura 2.6: (a) Immagine a livelli di grigio, (b) Immagine binaria cluster 1, (c) Immagine binaria cluster 2, (d) Immagine binaria cluster 3

L'immagine binaria (corrispondente ad esempio al cluster 1) data in input alla funzione sopracitata è una matrice il cui generico elemento (i, j) viene posto uguale a 1 se e solo se il pixel in posizione (r, s) nell'immagine di partenza apparteneva al cluster 1. Con la funzione *bwconncomp* di MATLAB è possibile trovare le componenti connesse (regioni di soli 1) dell'immagine binaria, successivamente, grazie alla funzione *regionprops* di MATLAB è possibile ottenere alcune misure delle proprietà di queste componenti connesse e in particolare le sottomatrici della matrice di input corrispondenti ad ogni componente connessa CC. Quindi, per ogni CC dell' immagine di partenza:

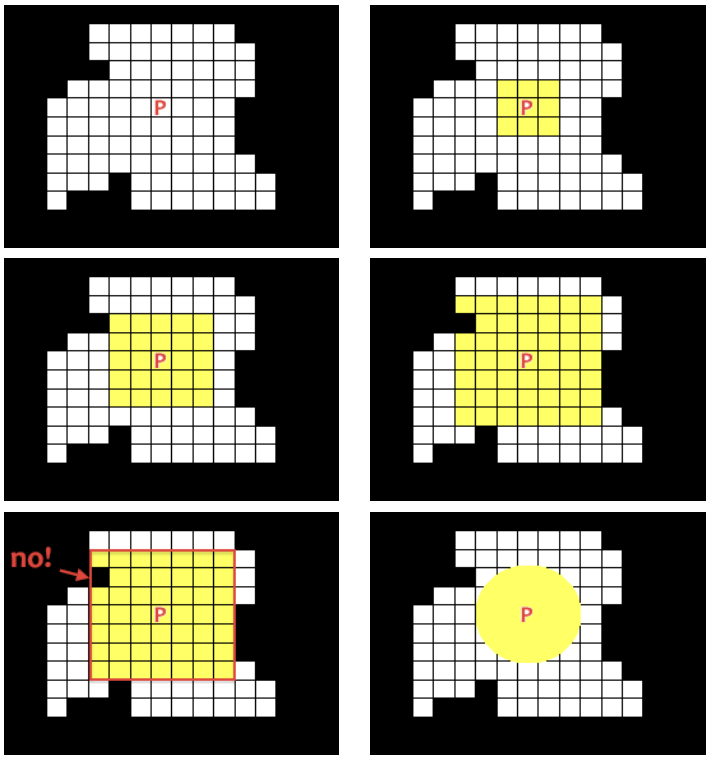


Figura 2.7: Esempio di collocamento di primitiva a partire da P

1. Per ogni pixel P di CC si cerca il cerchio di raggio maggiore centrato in P totalmente contenuto in CC: partendo da P si valuta se il quadrato di lato 3 di cui P è il pixel centrale è una matrice unitaria, se sì si ripete l'operazione precedente valutano il quadrato di lato 5 in cui P è il pixel centrale, e così via finché la matrice non contiene elementi nulli.

2. Una volta trovato, se il raggio è maggiore di 3, ne si salva l'area e il centro e il raggio rispettivamente nei vettori$AreaCerchiProvvisori$, $CentroCerchiProvvisori$ e $RaggiProvvisori$ (un contatore $ContaCerchiProvvisori$ stabilisce in che posizione scrivere il prossimo valore).

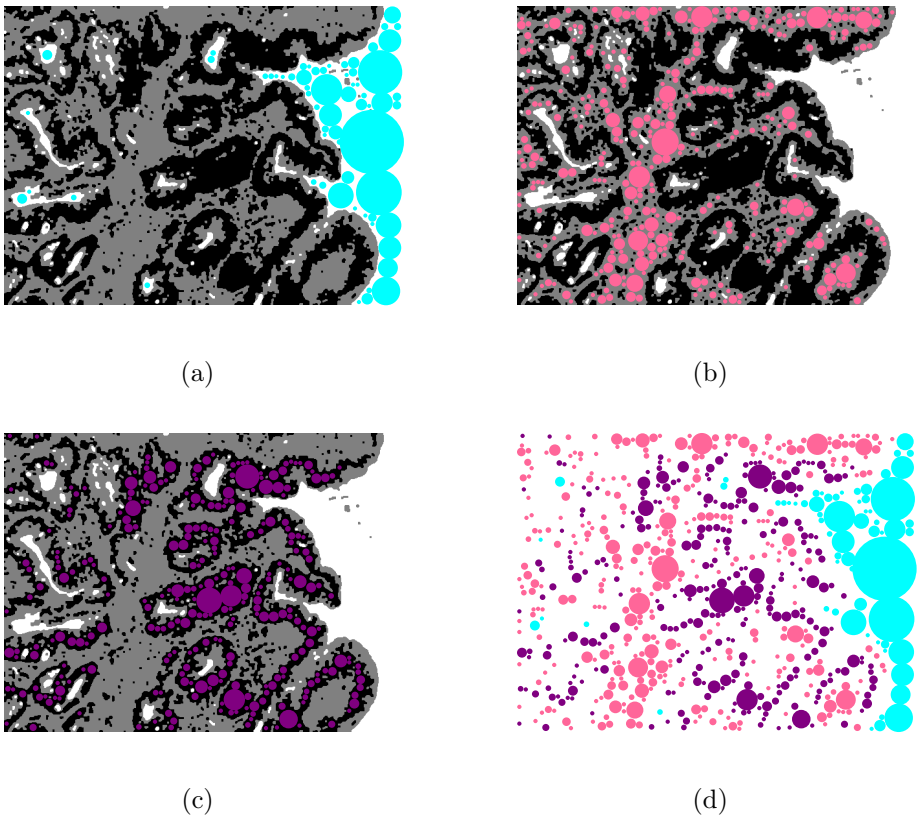
3. Valutati tutti i pixel di CC si cerca l'indice j tale che $AreaCerchiProvvisori(j) = max(AreaCerchiProvvisori)$. Si ispeziona la sottomatrice quadrata centrata in $CentroCerchiProvvisori(j)$ e di lato $2 * RaggiProvvisori(j)$: se in essa sono presenti solo pixel di valore unitario allora si salvano i valori $AreaCerchiProvvisori(j)$ e $CentroCerchiProvvisori(j)$ nei due vettori definitivi $Centri$ e $Aree$ e si considera questo stesso cerchio definitivo, altrimenti si azzerano i valori $AreaCerchiProvvisori(j)$ e $CentroCerchiProvvisori(j)$ e si torna al punto 3.

4. Si annullano i pixel della sottomatrice definita al punto 2.

5. Si azzerano$AreaCerchiProvvisori(j)$ e $CentroCerchiProvvisori(j)$.

6. Si ripetono tutti i passaggi a partire dal punto 3 finché il vettore *AreaCerchiProvvisori* risulta vuoto (la procedura è finita: sono state trovate tutte le primitive circolari che rappresentano CC).

Eseguendo questi passaggi per ogni cluster si ottengono le primitive circolari che lo rappresentano e infine, con la funzione *scatter* di MATLAB, è possibile plottare tutte le primitive del colore del cluster al quale appartengono.



(a) (b)

(c) (d)

Figura 2.8: (a) Primitive componente luminale, (b) Primitive componente stromale, (c) Primitive componente nucleare, (d) Immagine di primitive

2.2.3 Dall'immagine di primitive al grafo colorato

A partire dall'immagine di primitive si sceglie come insieme di nodi N del grafo G che si vuole costruire i centri dei cerchi definitivi precedentemente trovati. Per assegnare l'insieme di archi A al grafo G si costruisce una triangolazione di Delaunay a partire dall'insieme N utilizzando la funzione *DelaunayTri* di MATLAB.

Si definisce **triangolazione** una suddivisione dello spazio (2D nel nostro caso) tramite la connessione di tutti e soli punti dell'insieme N, in elementi triangolari tali che, l'intersezione degli elementi triangolari, presi a due a due sia un insieme vuoto, un vertice oppure un lato. Tra i possibili tipi di triangolazione, quella di Delaunay è univocamente determinata sull'insieme di punti N e gode delle seguente proprietà:

1. il cerchio circoscritto ad un generico elemento triangolare con vertici $v1, v2, v3$ non contiene alcun altro punto appartenente a N;

2. la triangolazione di Delaunay massimizza il minimo angolo interno dei triangoli.

Ottenuta la triangolazione DT è anche possibile ottenerne la collezione di archi ed effettuarne il plot con la funzione *triplot* di MATLAB.

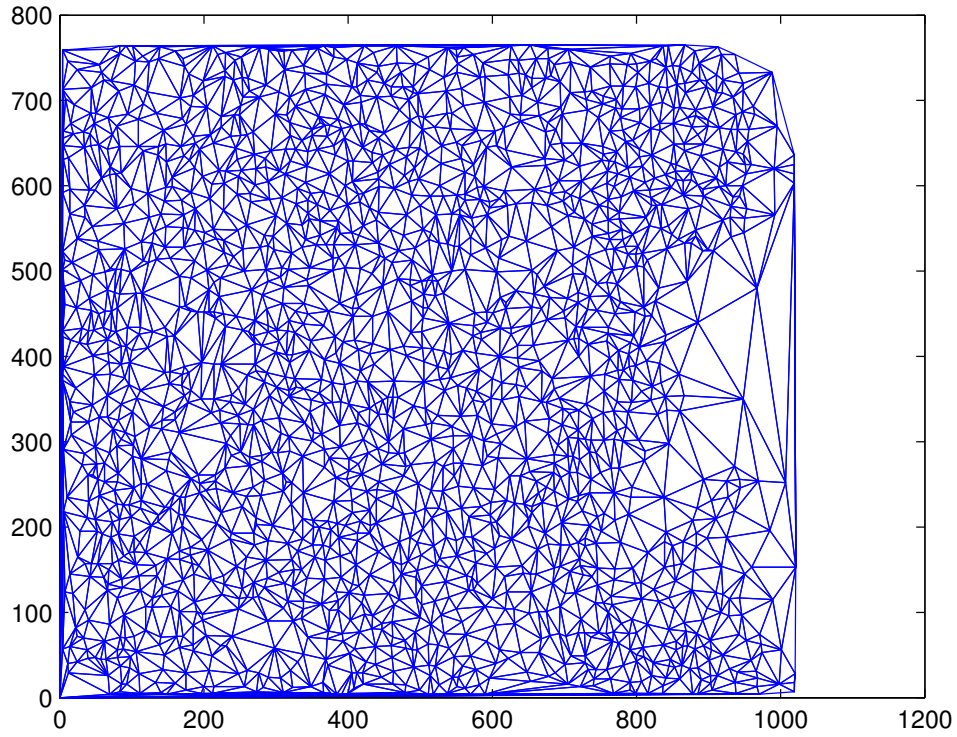


Figura 2.9: Grafo

Tuttavia il grafo così ottenuto non porta ancora alcuna informazione sulle relazioni spaziali esistenti fra le primitive dei diversi clusters. Per ottenere un grafo dotato di tali informazioni aggiuntive, ogni arco del grafo è stato colorato in base ai cluster di appartenenza dei suoi due nodi terminali. Essendo tre i cluster diversi, le possibili combinazioni di nodi presi a due a due sono sei, e quindi sono sei i possibili colori da attribuire ad ogni arco. In particolare, indicando con L un nodo appartenente alla componente luminale, con N un nodo appartenente alla componente nucleare e con S un nodo appartenente alla componente stromale e caratterizzando un arco in base ai sui nodi terminali, si ha:

- arco LL : rosso
- arco SS : giallo

- arco NN : nero
- arco LS : blu
- arco LN : verde
- arco SN : magenta

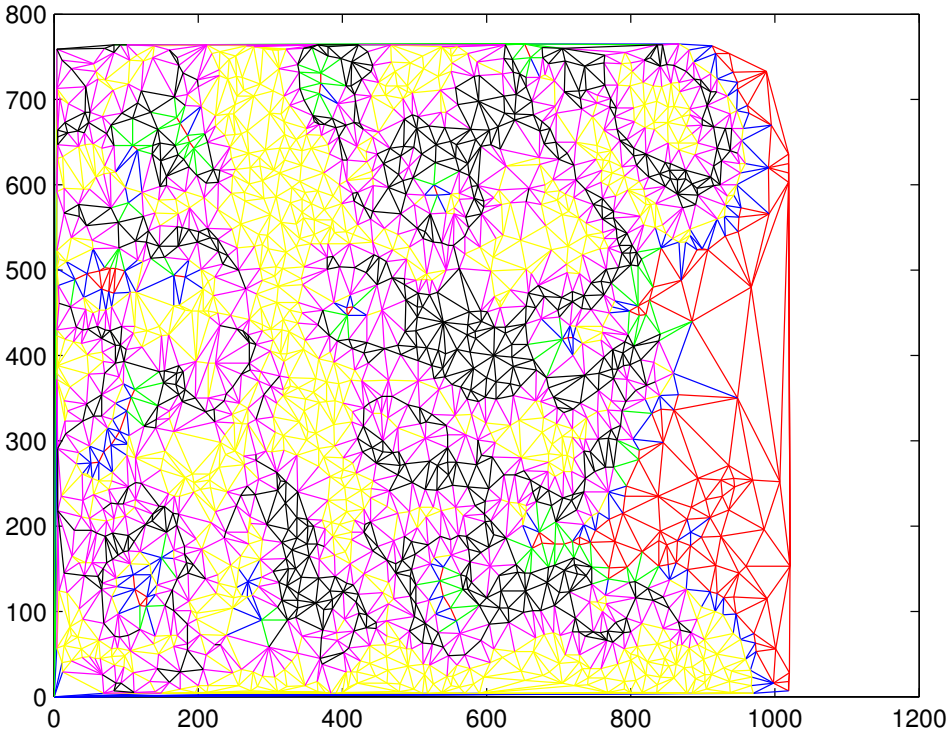


Figura 2.10: Grafo colorato

Capitolo 3

Estrazione delle features

In questo capitolo, dopo aver introdotto dei concetti generali sull'argomento della *feature estraction* si descriveranno le features estratte dal grafo colorato e il modo in cui sono state implementate.

3.1 Definizione di Feature

La *pattern recognition* è la disciplina che si occupa della classificazione di oggetti (patterns) in un dato numero di classi. In questo ambito, il termine inglese *features*, traducibile in italiano come "caratteristiche", sta ad indicare le misure usate per la classificazione. Le features sono cioè quello che viene normalmente definito come criterio di classificazione. In generale m features x_i con $i = 1, 2, ..., m$ formano il feature vector $X = [x_1, x_2, \ldots, x_m]$. E' importante notare che ogni pattern viene identificato da uno e un solo feature vector. Si definisce poi classificatore ciò che divide lo spazio dei feature vectors in regioni che corrispondono alla distinte classi mediante una regola di decisione.

3.2 Quantificazione del grafo colorato

In base a quanto detto in 2.1.2 e in base al procedimento spiegato in 3, è possibile fare due deduzioni importanti:

- Il grafo colorato è uno strumento che rappresenta in modo chiaro e sintetico le relazioni strutturali esistenti all'interno dell'immagine di partenza;

- Il grafo colorato è un oggetto dotato di strumenti, le sue proprietà globali, che ci consentono di quantificare il grado di "disordine" presente all'interno struttura ghiandolare.

Le proprietà che ci consentono di effettuare queste misure sono: il **grado medio**, il **coefficiente di clustering medio** e il **diametro**. Segue la descrizione di ciascuna di queste dal punto di vista teorico e implementativo.

3.3 Grado medio

3.3.1 Definizione

Si definisce grado di un nodo il numero di archi che da esso dipartono e quindi si definisce grado medio di un grafo la media aritmetica dei gradi di tutti i nodi del grafo. In questa prima feature tuttavia si è trascurata l'informazione derivante dal colore degli archi. E' possibile, infatti definire, ad esempio, grado rosso di un nodo il numero di archi rossi che da esso dipartono e conseguentemente grado medio rosso di un grafo la media aritmetica dei gradi rossi di tutti i nodi del grafo, e così per tutti i sei colori possibile. Alla fine quindi si avranno sette features per il grafo: grado medio, grado medio rosso, grado medio giallo, grado medio nero, grado medio blu, grado medio verde, grado medio magenta.

3.3.2 Implementazione

Si è creata una matrice $NumArchiPerNodo$ di dimensione $Nx7$ dove N è il numero di nodi del grafo e 7 è il numero gradi che si vogliono calcolare per ogni nodo. Per quanto riguarda ad esempio il primo nodo, si è contato il numero di archi che da esso uscivano (o entravano dato che il nostro grafo non è orientato) e lo si è memorizzato in $NumArchiPerNodo(1,1)$: contemporaneamente si è ispezionato il colore di tali archi (informazione memorizzata nel vettore $ColoreArchi$) e l'informazione ricavata è stata salvata nelle caselle $NumArchiPerNodo(1, 2:7)$. Infine si è calcolata la media aritmetica dei vari gradi, ottenendo così le features $GraphAvDegree$, $GraphAvDegreeRED$, $GraphAvDegreeYELLOW$, $GraphAvDegreeBLACK$, $GraphAvDegreeBLUE$, $GraphAvDegreeGREEN$ e $GraphAvDegreeMAGENTA$.

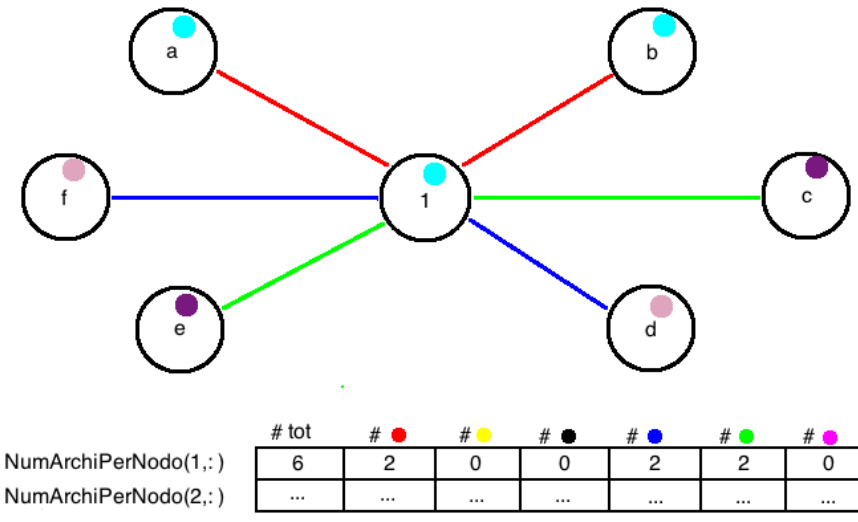


Figura 3.1: Grado di un nodo

3.4 Coefficiente di clustering medio

3.4.1 Definizione

Il coefficiente di clustering CC_i di un nodo i si definisce

$$CC_i = \frac{2E_i}{(d_i(d_i - 1))}$$

dove d_i è il numero di nodi adiacenti ad i mentre E_i è il numero di archi esistenti fra questi nodi adiacenti. Il coefficiente di clustering permette quindi di quantificare il grado di connettività nella componente connessa nelle vicinanze del nodo i. Nel grafo colorato è possibile individuare quattro coefficienti di clustering medi. Il primo è coefficiente di clustering medio incolore che consiste nella media di tutti i CC_i sopra definiti calcolato escludendo i nodi che hanno grado 0 o 1 (valori che annullano il denominatore). Per calcolare i successivi tre coefficienti di clustering medi è necessario definire per ogni nodo i appartenente al cluster J il coefficiente di clustering

$$CC_i(J) = \frac{2E(J)}{d_i(J)(d_i(J) - 1)}$$

dove $d_i(J)$ è il numero di archi che connettono il nodo i ad altri nodi del cluster J (archi JJ), mentre $E(J)$ è il numero di archi JJ esistenti fra questi nodi. Calcolando,per ogni cluster J, la media di tutti i $CC_i(J)$, si ottengono i tre coefficienti di clustering colorati.

3.4.2 Implementazione

E' stata innanzitutto creata una matrice $CCNodi$ di dimensione $Nx4$ dove N è il numero di nodi del grafo e 4 è, nel nostro caso, il numero di tipi di coefficienti di clustering CC che vogliamo calcolare: CC (incolore), $CCred$, $CCyellow$ e $CCblack$. In particolare, ogni posizione del tipo $(i, 1)$ di '$CCNodi$' tiene conto di tutti gli archi esistenti fra nodi connessi al nodo n, mentre le posizioni $(i, 2 : 4)$ di CCNodi valutano rispettivamente solo gli archi rossi, gialli e neri. Successivamente per ogni nodo n, scorrendo il vettore $edges$ sono stati salvati nel vettore '$IndiciNodoInVettoreArchi$' tutti gli indici j del vettore degli archi tali che $edges(j, :) = (x, n)$ oppure $edges(j, :) = (n, x)$. In questo modo il d_n definito nel paragrafo precedente è esattamente la lunghezza del vettore '$IndiciNodoInVettoreArchi$'. Per calcolare E_n, posto inizialmente uguale a d_n (dato che almeno d_i archi sono sicuramente compresi), è stato possibile verificare per ogni nodo adiacente al nodo n la presenza di archi che lo collegassero ad altri nodi adiacenti al nodo n di partenza ed in caso affermativo è stato incrementato di una unità il valore di E_n. Seguendo le definizioni del paragrafo precedente si sono infine ottenuti: $GraphAvCC$,$GraphAvCCRED$,$GraphAvCCYELLOW$ e $GraphAvCCBLACK$.

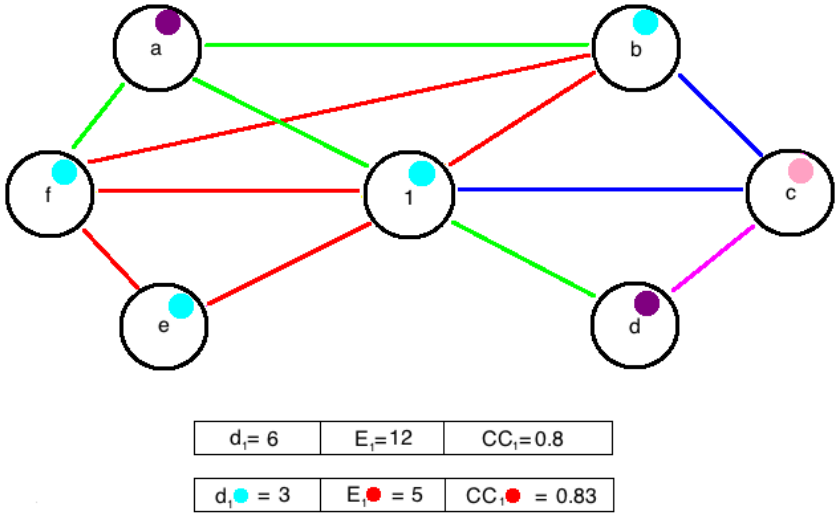


Figura 3.2: Esempio di calcolo del coefficiente di clustering (incolore e rosso)

3.5 Diametro

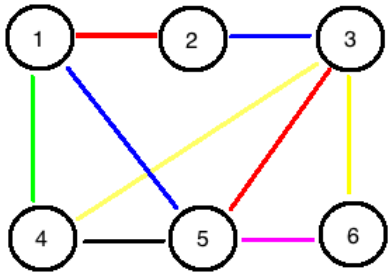
3.5.1 Definizione

Definiamo diametro incolore di un grafo il percorso minimo più lungo fra qualsiasi coppia di nodi del grafo. Nel caso del grafo colorato, è possibile definire altri sei tipi di diametro, dove questi sono calcolati considerando, di volta in volta, solamente gli archi del colore di interesse.

3.5.2 Implementazione

Per il calcolo di queste sette features è stato utilizzato il *Bioinformatics Toolbox* di MATLAB, poiché questo risultava già dotato della funzioni utili alla generazione di grafi e al calcolo dei cammini minimi. E' stata innanzitutto definita una matrice di adiacenze '$CMatrix'$ di dimensione NxN, dove N è il numero di nodi del grafo, i cui generici elementi (i, j) e (j, i) sono stati posti uguale a 1 se e solo se almeno una fra (i, j) e (j, i) è una coppia appartenente alla collazione degli archi del grafo di partenza. A partire da $CMatrix$ è stato creato l'oggetto $BGobj$ e successivamente, grazie alla funzione *allshortestpaths*, è stato possibile ottenere nella matrice *dist* la lunghezza di tutti i cammini minimi per ogni coppia di nodi del grafo. Ispezionando la matrice *dist* e operando un controllo di tipo $isfinite$ (per evitare i casi degeneri derivanti da distanze infinite) è stato possibile trovare il massimo dei cammini minimi: il diametro incolore. Per ottenere invece i diametri colorati è stata definita per ogni colore una matrice $CMatrixCOLORE$ in cui gli elementi generici (i, j) e (j, i) sono stati posti uguali a 1 se e solo se almeno una fra (i, j) e (j, i) è una coppia appartenete alla collezione degli archi il cui colore è COLORE. Da questo punto in poi il procedimento è invariato rispetto al precedente e consente di ottenere le

features: $GraphDiameterRED$, $GraphDiameterYellow$, $GraphDiameterBlack$, $GraphDiameterBlue$, $GraphDiameterGREEN$ e $GraphDiameterMAGENTA$.



CMatrix

	1	2	3	4	5	6
1	0	1	0	1	1	0
2	1	0	1	0	0	0
3	0	1	0	1	1	1
4	1	0	1	0	1	0
5	1	0	1	1	0	1
6	0	0	1	0	1	0

CMatrixRed

	1	2	3	4	5	6
1	0	1	0	0	0	0
2	1	0	0	0	0	0
3	0	0	0	0	1	0
4	0	0	0	0	0	0
5	0	0	1	0	0	0
6	0	0	0	0	0	0

Figura 3.3: Esempio di matrici di adiacenza

Capitolo 4

Conclusioni

In questo ultimo capitolo sono raccolti i risultati ottenuti applicando il programma a tre immagini istologiche di epitelio esofageo. Le tre immagini a disposizione sono tre diversi ingrandimenti dello stesso tessuto. Da ora in poi si farà riferimento alle tre immagini con i nomi *Medio Ingrandimento*, *Medio Ingrandimento Bis* e *Forte Ingrandimento*. Una volta caricata l'immagine è stata innanzitutto riscalata (si veda A, riga 6): tale passaggio si è rivelato imprescindibile in quanto l'immagine iniziale risultava altresì eccessivamente grande (1536x2048 pixels) per essere processata in un tempo contenuto. Si vuole però qui sottolineare che ***tale scelta non dovrebbe comportare, almeno in linea teorica, significative perdite di informazione poiché l'approccio usato è di tipo strutturale e quindi non basato sui singoli pixels*** ma sulla distribuzione spaziale dei componenti cellulari.

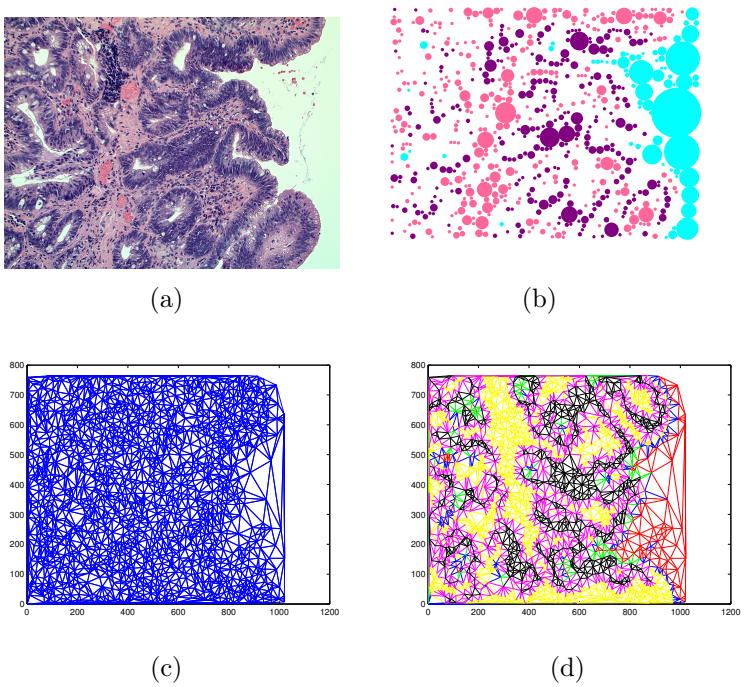
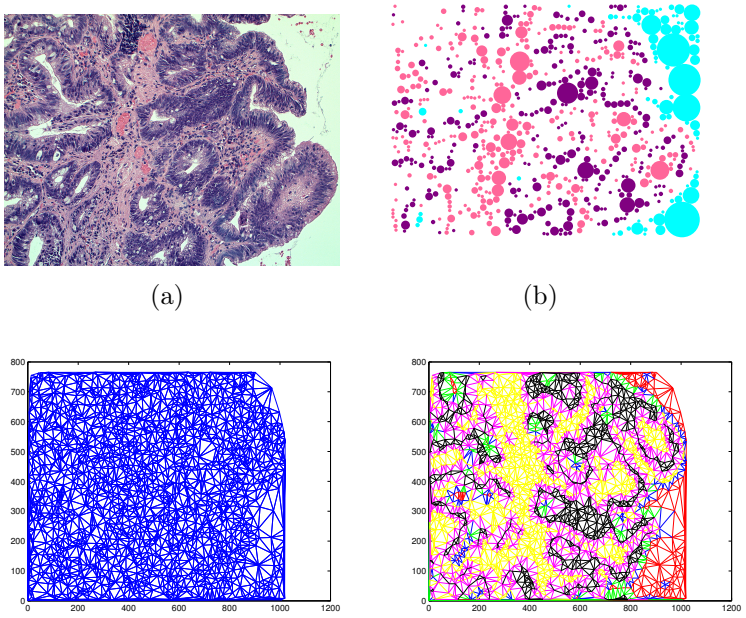


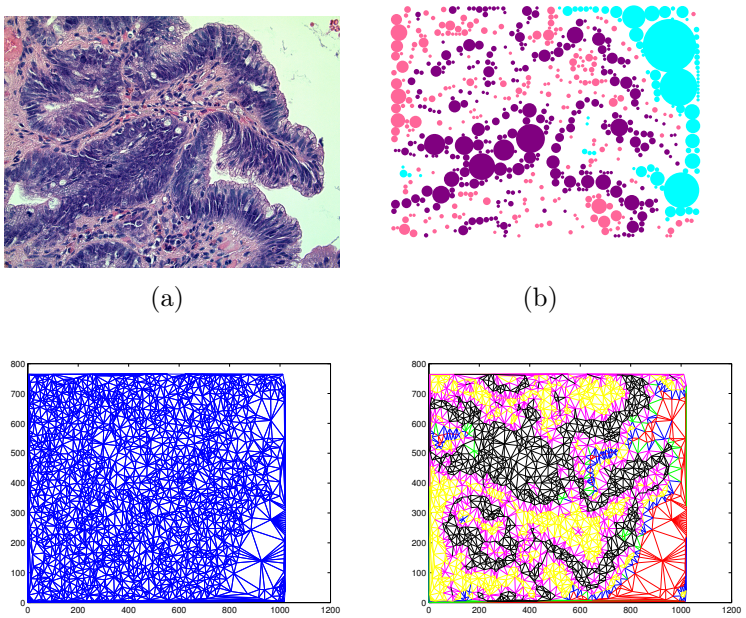
Figura 4.1: Stadi dell'elaborazione dell'immagine Medio Ingrandimento: (a) Immagine originale, (b) Immagine di primitive, (c) Grafo, (d) Grafo colorato



(a) (b)

(c) (d)

Figura 4.2: Stadi dell'elaborazione dell'immagine Medio Ingrandimento Bis: (a) Immagine originale, (b) Immagine di primitive, (c) Grafo, (d) Grafo colorato



(a) (b)

(c) (d)

Figura 4.3: Stadi dell'elaborazione dell'immagine Forte Ingrandimento: (a) Immagine originale, (b) Immagine di primitive, (c) Grafo, (d) Grafo colorato

La seguente tabella riporta i valori ottenuti calcolando le *18 features* a partire dai grafi colorati delle rispettive immagini di partenza:

Feature:	*MedioIngr.*	*MedioIngr.Bis*	*ForteIngr.*
GraphAvCC	19,9573	19,8777	19,893
GraphAvCCBLACK	1,1503	1,1186	1,858
GraphAvCCRED	0,2288	0,3064	0,2261
GraphAvCCYELLOW	1,6127	1,5447	1,2403
GraphAvDegree	5,9826	5,9713	5,9705
GraphAvDegreeBLACK	1,6035	1,5564	2,2119
GraphAvDegreeBLUE	0,2789	0,2734	0,2731
GraphAvDegreeGREEN	0,1656	0,216	0,0874
GraphAvDegreeMAGENTA	1,5959	1,5532	1,4265
GraphAvDegreeRED	0,3072	0,4064	0,3113
GraphAvDegreeYELLOW	2,0316	1,966	1,6603
GraphDiameter	26	28	26
GraphDiameterBLACK	60	62	69
GraphDiameterBLUE	31	39	20
GraphDiameterGREEN	14	12	10
GraphDiameterMAGENTA	98	107	80
GraphDiameterRED	18	25	19
GraphDiameterYELLOW	47	57	63

In conclusione si vuole porre in evidenza che i risultati fin qui descritti costituiscono uno studio pilota atto a verificare che il metodo proposto in [**1**] possa essere applicabile anche alle immagini istologiche di tessuto esofageo. Essendo i risultati qualitativi soddisfacenti, il procedimento proposto potrebbe essere applicato ad un dataset più vasto e potrebbe inoltre costituire un punto di partenza per eventuali fasi successive di classificazione o segmentazione. Per quanto riguarda eventuali migliorie da apportare al programma, bisognerebbe sicuramente trovare un'alternativa alla generazione del grafo colorato proposta in questo elaborato: tale porzione di programma risulta molto gravosa da un punto di vista prestazionale.

Appendice A

A.1 Programma

```
1
close all
3 clear all

5 R=input('Inserire nome del file con estensione: ','s');
imtest=im2double(imread(R));
7 imtest=imresize(imtest,0.3,'bilinear');
% l'immagine è stata scalata per motivi di complessità di calcolo
9
figure
11 title('immagine originale');
imshow(imtest);
13

15 % Clustering con kmeans

17 cform=makecform('srgb2lab');      % Si passa nel dominio L*a*b (
    vedi 2.2.1)
lab_imtest=applycform(imtest,cform);
19 ab=double(lab_imtest(:,:,2:3));
nrows=size(ab,1);
21 ncols=size(ab,2);
ab=reshape(ab,nrows*ncols,2);
23 ncolors=3;
[indice,centroid]=kmeans(ab,ncolors,'distance','sqEuclidean','
    Replicates',3);
25 pixel_labels=reshape(indice,nrows,ncols);

27 for ct = 1:3
    eval(['nn', num2str(ct), ' = find(pixel_labels==', num2str(ct),
        ');'])
29     eval(['NumPix', num2str(ct), '=length(nn', num2str(ct), ');'])
    eval(['SommaIntens', num2str(ct), '= sum(imtest(nn', num2str(ct
        ),'));'])
31     eval(['IntensMedia', num2str(ct), '=SommaIntens', num2str(ct),
        '/NumPix', num2str(ct),';'])
end
33
vettoreIntensitaMedie = [IntensMedia1,IntensMedia2,IntensMedia3];
35
```

```
Azzurro = max(vettoreIntensitaMedie);
Viola = min(vettoreIntensitaMedie);
Rosa = (SommaIntens1/NumPix1+SommaIntens2/NumPix2+SommaIntens3/
    NumPix3)- Azzurro - Viola;

% Mappa colorata e a livelli di grigio per passare dalla matrice
    di indici all'immagine quantizzata

if IntensMedia1==Azzurro
    if IntensMedia2==Viola
        map=[0 1 1;128/255 0 128/255; 1 0.4 0.6];
        im1=ind2rgb(pixel_labels,map);
        map2=[1 1 1; 0 0 0; 0.5 0.5 0.5];
    elseif IntensMedia3==Viola
        map=[0 1 1;1 0.4 0.6 ;128/255 0 128/255];
        im1=ind2rgb(pixel_labels,map);
        map2=[1 1 1;0.5 0.5 0.5;0 0 0];
    end
elseif IntensMedia2==Azzurro
    if IntensMedia3==Viola
        map=[1 0.4 0.6;0 1 1;128/255 0 128/255];
        im1=ind2rgb(pixel_labels,map);
        map2=[0.5 0.5 0.5;1 1 1;0 0 0];
    elseif IntensMedia1==Viola
        map=[128/255 0 128/255;0 1 1;1 0.4 0.6];
        im1=ind2rgb(pixel_labels,map);
        map2=[0 0 0;1 1 1;0.5 0.5 0.5];
    end
elseif IntensMedia3==Azzurro
    if IntensMedia2==Viola
        map=[1 0.4 0.6;128/255 0 128/255;0 1 1];
        im1=ind2rgb(pixel_labels,map);
        map2=[0.5 0.5 0.5;0 0 0;1 1 1];
    elseif IntensMedia1==Viola
        map=[128/255 0 128/255;1 0.4 0.6;0 1 1];
        im1=ind2rgb(pixel_labels,map);
        map2=[0 0 0;0.5 0.5 0.5;1 1 1];
    end
end

immagine_grayscale=ind2gray(pixel_labels,map2);
immagine_grayscale=medfilt2(medfilt2(medfilt2(immagine_grayscale))
    );
se1 = strel('square',3);
im3=imopen(imclose(immagine_grayscale,se1),se1);

immagine_opened_closed=zeros(size(immagine_grayscale,1),size(
    immagine_grayscale,2),3);

for i=1:size(im3,1)
    for j=1:size(im3,2)
        if im3(i,j)==0
            immagine_opened_closed(i,j,1)=128/255; %viola
            immagine_opened_closed(i,j,2)=0;
            immagine_opened_closed(i,j,3)=128/255;
        elseif im3(i,j)==0.5
```

```
            immagine_opened_closed(i,j,1)=1; % rosa
            immagine_opened_closed(i,j,2)=0.4;
            immagine_opened_closed(i,j,3)=0.6;
        else
            immagine_opened_closed(i,j,1)=0; % azzurro
            immagine_opened_closed(i,j,2)=1;
            immagine_opened_closed(i,j,3)=1;
        end
    end
end

imm_cluster_1a = im3==1;  % immagine binaria cluster 1
imm_cluster_2a = im3==0.5; % immagine binaria cluster 2
imm_cluster_3a = im3==0; % immagine binaria cluster 3


% Allocazione primitive circolari su imm_cluster1
[Centri_1,Raggi_1,Aree_1] = prim_allocation(imm_cluster_1a);

% Allocazione primitive circolari su imm_cluster2
[Centri_2,Raggi_2,Aree_2] = prim_allocation(imm_cluster_2a);

% Allocazione primitive circolari su imm_cluster3
[Centri_3,Raggi_3,Aree_3] = prim_allocation(imm_cluster_3a);


figure(1);
imshow(im3);
hold on
tmpCen = sum(Centri_1,2);
tmpN = find(tmpCen>0,1,'last');
scatter(Centri_1(1:tmpN,1),Centri_1(1:tmpN,2),Aree_1(1:tmpN),'c','
    filled'); % stampa primitive cluster 1
hold on
tmpCen = sum(Centri_2,2);
tmpN = find(tmpCen>0,1,'last');
scatter(Centri_2(1:tmpN,1),Centri_2(1:tmpN,2),Aree_2(1:tmpN),[1
    0.4 0.6],'filled'); % stampa primitive cluster 2
hold on
tmpCen = sum(Centri_3,2);
tmpN = find(tmpCen>0,1,'last');
scatter(Centri_3(1:tmpN,1),Centri_3(1:tmpN,2),Aree_3(1:tmpN)
    ,[128/255 0 128/255],'filled');% stampa primitive cluster 3

X1=Centri_1(:,1);
Y1=Centri_1(:,2);
XY1=[X1(:,1:end),Y1(:,1:end)];

X2=Centri_2(:,1);
Y2=Centri_2(:,2);
XY2=[X2(:,1:end),Y2(:,1:end)];

X3=Centri_3(:,1);
Y3=Centri_3(:,2);
XY3=[X3(:,1:end),Y3(:,1:end)];
```

```
141
dt = DelaunayTri([X1;X2;X3],[Y1;Y2;Y3]);
143 edges=edges(dt); % restituisce tutti gli archi della
    triangolazione

145 % Grafo senza informazione sui colori
figure(2);
147 triplot(dt);

149 % Prima colonna di matrCol è num identificativo del colore, ad es:
     1->[1 0 0]
matrCol = [1 1 0 0;
151            2 1 1 0;
               3 0 0 0;
153            4 0 0 1;
               5 0 1 0;
155            6 1 0 1];

157 % ColoreArchi memorizza per ogni arco il num identificativo del
    colore corrispondente
ColoreArchi=zeros(length(edges),1);
159
for i=1:length(edges)
161
    PrimoNodo=dt.X(edges(i,1),:);
163     SecondoNodo=dt.X(edges(i,2),:);

165     if ismember(PrimoNodo,XY1,'rows')==1
        if ismember(SecondoNodo,XY1,'rows')==1
167         ColoreArchi(i)=1; %red
        elseif ismember(SecondoNodo,XY2,'rows')==1
169         ColoreArchi(i)=4;%blue
        elseif ismember(SecondoNodo,XY3,'rows')==1
171             ColoreArchi(i)=5;%green
        end
173
    elseif ismember(PrimoNodo,XY2,'rows')==1
175    if ismember(SecondoNodo,XY2,'rows')==1
           ColoreArchi(i)=2;%yellow
177    elseif ismember(SecondoNodo,XY3,'rows')==1
           ColoreArchi(i)=6;%magenta
179    elseif ismember(SecondoNodo,XY1,'rows')==1
            ColoreArchi(i)=4;%blue
181    end

183     elseif ismember(PrimoNodo,XY3,'rows')==1
       if ismember(SecondoNodo,XY3,'rows')==1
185         ColoreArchi(i)=3;%black
       elseif ismember(SecondoNodo,XY1,'rows')==1
187         ColoreArchi(i)=5;%green
       elseif ismember(SecondoNodo,XY2,'rows')==1
189          ColoreArchi(i)=6;%magenta
       end
191     end
end
193
```

```
% plot del grafo colorato
figure(3)

for i=1:length(edges)
plot([dt.X(edges(i,1),1) dt.X(edges(i,2),1)],[dt.X(edges(i,1),2)
    dt.X(edges(i,2),2)],'Color',matrCol(ColoreArchi(i),2:4));
hold on
end


%       FEATURE 1: Graph Average Degree                 %


% Descrizione matrice NumArchiPerNodo:
%
% prima colonna: tutti archi (indip dal colore),
% seconda colonna: solo archi rossi,
% terza colonna: solo archi gialli,
% etc ...

NumArchiPerNodo=zeros(length(dt.X),7);

for i=1:length(dt.X)

    % IndiciNodoInVettArchi raccoglie gli indici di tutti gli
       archi di edges che dipartono da ogni nodo!
    IndiciNodoInVettArchi=find(edges==i);

    NumArchiPerNodo(i,1)=length(IndiciNodoInVettArchi);

% dt non ha orientazione ma la matrice edges non presenta
   ripetizioni (se esiste edges(i)=[49 25] non          esiste
   anche edges(j)=[25 49]) e quindi il find va fatto su entrambe
   le colonne di edges.

    for j=1:length(IndiciNodoInVettArchi)
        if IndiciNodoInVettArchi(j)<=length(edges)
           IndiciNodoInVettArchi(j)=IndiciNodoInVettArchi(j);
        else
            IndiciNodoInVettArchi(j)=IndiciNodoInVettArchi(j)-
               length(edges);
        end
    end

    for k=1:length(IndiciNodoInVettArchi)

      % conteggia archi in base al colore:

      if ColoreArchi(IndiciNodoInVettArchi(k))==1
         NumArchiPerNodo(i,2)=NumArchiPerNodo(i,2)+1;
      elseif ColoreArchi(IndiciNodoInVettArchi(k))==2
         NumArchiPerNodo(i,3)=NumArchiPerNodo(i,3)+1;
      elseif ColoreArchi(IndiciNodoInVettArchi(k))==3
         NumArchiPerNodo(i,4)=NumArchiPerNodo(i,4)+1;
      elseif ColoreArchi(IndiciNodoInVettArchi(k))==4
         NumArchiPerNodo(i,5)=NumArchiPerNodo(i,5)+1;
```

```
      elseif ColoreArchi(IndiciNodoInVettArchi(k))==5
245       NumArchiPerNodo(i,6)=NumArchiPerNodo(i,6)+1;
      elseif ColoreArchi(IndiciNodoInVettArchi(k))==6
247       NumArchiPerNodo(i,7)=NumArchiPerNodo(i,7)+1;
      end
249
    end
251 end

253 GraphAvDegree=sum(NumArchiPerNodo(:,1))/length(dt.X);
    GraphAvDegreeRED=sum(NumArchiPerNodo(:,2))/length(dt.X);
255 GraphAvDegreeYELLOW=sum(NumArchiPerNodo(:,3))/length(dt.X);
    GraphAvDegreeBLACK=sum(NumArchiPerNodo(:,4))/length(dt.X);
257 GraphAvDegreeBLUE=sum(NumArchiPerNodo(:,5))/length(dt.X);
    GraphAvDegreeGREEN=sum(NumArchiPerNodo(:,6))/length(dt.X);
259 GraphAvDegreeMAGENTA=sum(NumArchiPerNodo(:,7))/length(dt.X);

261
    %      FEATURE 2: Average Clustering Coefficient       %
263


265
    % Descrizione vettore ClusteringCoefficientsNodi:
267 %
    % prima colonna: CC che tiene conto di tutti gli archi,
269 % seconda,Terza e Quarta colonna: CC che tiene conto degli archi "
      corrispondenti" al cluster di appartenenza

271 CCNodi=zeros(length(dt.X),4);

273 for i=1:length(dt.X)

275     IndiciNodoInVettArchi=find(edges==i);

277     for j=1:length(IndiciNodoInVettArchi)
          if IndiciNodoInVettArchi(j)<=length(edges)
279           IndiciNodoInVettArchi(j)=IndiciNodoInVettArchi(j);
          else
281           IndiciNodoInVettArchi(j)=IndiciNodoInVettArchi(j)-
                length(edges);
          end
283     end


285
        % d=#vicini
287     d=length(IndiciNodoInVettArchi);

289     % E=#archi esistenti fra i vicini
        E=length(IndiciNodoInVettArchi);
291
        if ismember(dt.X(i,:),XY1,'rows')==1
293         col=1;
        elseif ismember(dt.X(i,:),XY2,'rows')==1
295         col=2;
        elseif ismember(dt.X(i,:),XY3,'rows')==1
297         col=3;
```

```
    end

    for j=1:length(IndiciNodoInVettArchi)
        Neighbor1=edges(IndiciNodoInVettArchi(j),1)+edges(
            IndiciNodoInVettArchi(j),2)-i;

        for k=1:length(IndiciNodoInVettArchi)
            if k~=j
             Neighbor2=edges(IndiciNodoInVettArchi(k),1)+edges(
                 IndiciNodoInVettArchi(k),2)-i;
             if ismember([Neighbor1 Neighbor2],edges,'rows')==1
             E=E+1;
             [~,~,index]=intersect([Neighbor1 Neighbor2],edges,'
                 rows');

                if ColoreArchi(index)==col
                    CCNodi(i,col+1)=CCNodi(i,col+1)+1;
                end

             end
            end
        end
    end

    if d>1
    CCNodi(i,1)=(2*E)/d*(d-1);
    end

end

GraphAvCC=sum(CCNodi(:,1))/length(dt.X);
GraphAvCCRED=sum(CCNodi(:,2))/length(dt.X);
GraphAvCCYELLOW=sum(CCNodi(:,3))/length(dt.X);
GraphAvCCBLACK=sum(CCNodi(:,4))/length(dt.X);


%                 FEATURE 3: Diameter                          %


CMatrix=zeros(length(dt.X));

for i=1:length(dt.X)
    for j=1:length(dt.X)
        if ismember([i j],edges,'rows')==1
            CMatrix(i,j)=1;
            CMatrix(j,i)=1;
        end
    end
end


BGobj = biograph(CMatrix);

for i=1:length(edges)
set(BGobj.Edges(i), 'LineColor',matrCol(ColoreArchi(i),2:4),'
    Weight', 1);
```

```
end

for i=1:length(dt.X)
set(BGobj.Nodes(i), 'Position',dt.X(i,:));
end

[dist] = allshortestpaths(BGobj);

GraphDiameter=0;

for  i=1:size(dist,1)
   for j=1:size(dist,2)
        if isfinite(dist(i,j))==1
            if dist(i,j)>GraphDiameter
                GraphDiameter=dist(i,j);
            end
        end
    end
end

% DiameterRED

CMatrixRED=zeros(length(dt.X));

for i=1:length(dt.X)
    for j=1:length(dt.X)

        if ismember([i j],edges,'rows')==1&&ismember([dt.X(i,1) dt
           .X(i,2)],XY1,'rows')&&ismember([dt.X(j,1) dt.X(j,2)],
           XY1,'rows')
            CMatrixRED(i,j)=1;
            CMatrixRED(j,i)=1;
        end
    end
end


BGobjRED = biograph(CMatrixRED);

for i=1:nnz(CMatrixRED)
set(BGobjRED.Edges(i),'Weight', 1);
end



[distRED] = allshortestpaths(BGobjRED);

GraphDiameterRED=0;

for  i=1:size(distRED,1)
   for j=1:size(distRED,2)
        if isfinite(distRED(i,j))==1
            if distRED(i,j)>GraphDiameterRED
                GraphDiameterRED=distRED(i,j);
            end
        end
```

```
    end
end

% DiameterYELLOW

CMatrixYELLOW=zeros(length(dt.X));

for i=1:length(dt.X)
    for j=1:length(dt.X)

        if ismember([i j],edges,'rows')==1&&ismember([dt.X(i,1) dt
            .X(i,2)],XY2,'rows')&&ismember([dt.X(j,1) dt.X(j,2)],
            XY2,'rows')
             CMatrixYELLOW(i,j)=1;
             CMatrixYELLOW(j,i)=1;
        end
    end
end


BGobjYELLOW = biograph(CMatrixYELLOW);

for i=1:nnz(CMatrixYELLOW)
set(BGobjYELLOW.Edges(i),'Weight', 1);
end


[distYELLOW] = allshortestpaths(BGobjYELLOW);

GraphDiameterYELLOW=0;

for  i=1:size(distYELLOW,1)
   for j=1:size(distYELLOW,2)
        if isfinite(distYELLOW(i,j))==1
            if distYELLOW(i,j)>GraphDiameterYELLOW
                GraphDiameterYELLOW=distYELLOW(i,j);
            end
        end
    end
end

% DiameterBLACK

CMatrixBLACK=zeros(length(dt.X));

for i=1:length(dt.X)
    for j=1:length(dt.X)

        if ismember([i j],edges,'rows')==1&&ismember([dt.X(i,1) dt
            .X(i,2)],XY3,'rows')&&ismember([dt.X(j,1) dt.X(j,2)],
            XY3,'rows')
             CMatrixBLACK(i,j)=1;
             CMatrixBLACK(j,i)=1;
        end
    end
end
```

```
457
BGobjBLACK = biograph(CMatrixBLACK);
459
for i=1:nnz(CMatrixBLACK)
461 set(BGobjBLACK.Edges(i),'Weight', 1);
end
463


465
[distBLACK] = allshortestpaths(BGobjBLACK);
467
GraphDiameterBLACK=0;
469
for  i=1:size(distBLACK,1)
471    for j=1:size(distBLACK,2)
         if isfinite(distBLACK(i,j))==1
473          if distBLACK(i,j)>GraphDiameterBLACK
               GraphDiameterBLACK=distBLACK(i,j);
475          end
         end
477     end
end
479
% DiameterBLUE
481
CMatrixBLUE=zeros(length(dt.X));
483
for i=1:length(dt.X)
485     for j=1:length(dt.X)

487          if ismember([i j],edges,'rows')==1&&ismember([dt.X(i,1) dt
             .X(i,2)],XY1,'rows')&&ismember([dt.X(j,1) dt.X(j,2)],
             XY2,'rows')
              CMatrixBLUE(i,j)=1;
489           CMatrixBLUE(j,i)=1;
          end
491     end
end
493


495 BGobjBLUE = biograph(CMatrixBLUE);

497 for i=1:nnz(CMatrixBLUE)
    set(BGobjBLUE.Edges(i),'Weight', 1);
499 end


501


503 [distBLUE] = allshortestpaths(BGobjBLUE);

505 GraphDiameterBLUE=0;

507 for  i=1:size(distBLUE,1)
    for j=1:size(distBLUE,2)
509      if isfinite(distBLUE(i,j))==1
```

```
            if distBLUE(i,j)>GraphDiameterBLUE
511                 GraphDiameterBLUE=distBLUE(i,j);
            end
513         end
    end
515 end

517 % DiameterGREEN

519 CMatrixGREEN=zeros(length(dt.X));

521 for i=1:length(dt.X)
    for j=1:length(dt.X)
523
        if ismember([i j],edges,'rows')==1&&ismember([dt.X(i,1) dt
            .X(i,2)],XY1,'rows')&&ismember([dt.X(j,1) dt.X(j,2)],
            XY3,'rows')
525          CMatrixGREEN(i,j)=1;
             CMatrixGREEN(j,i)=1;
527     end
    end
529 end


531
BGobjGREEN = biograph(CMatrixGREEN);
533
for i=1:nnz(CMatrixGREEN)
535 set(BGobjGREEN.Edges(i),'Weight', 1);
end
537



539
[distGREEN] = allshortestpaths(BGobjGREEN);
541
GraphDiameterGREEN=0;
543
for  i=1:size(distGREEN,1)
545   for j=1:size(distGREEN,2)
        if isfinite(distGREEN(i,j))==1
547         if distGREEN(i,j)>GraphDiameterGREEN
                GraphDiameterGREEN=distGREEN(i,j);
549         end
        end
551 end
end
553
% DiameterMAGENTA
555
CMatrixMAGENTA=zeros(length(dt.X));
557
for i=1:length(dt.X)
559 for j=1:length(dt.X)

561     if ismember([i j],edges,'rows')==1&&ismember([dt.X(i,1) dt
            .X(i,2)],XY2,'rows')&&ismember([dt.X(j,1) dt.X(j,2)],
            XY3,'rows')
```

```
            CMatrixMAGENTA(i,j)=1;
563         CMatrixMAGENTA(j,i)=1;
        end
565     end
end
567

569 BGobjMAGENTA = biograph(CMatrixMAGENTA);

571 for i=1:nnz(CMatrixMAGENTA)
set(BGobjMAGENTA.Edges(i),'Weight', 1);
573 end


575

577 [distMAGENTA] = allshortestpaths(BGobjMAGENTA);

579 GraphDiameterMAGENTA=0;

581 for  i=1:size(distMAGENTA,1)
   for j=1:size(distMAGENTA,2)
583         if isfinite(distMAGENTA(i,j))==1
              if distMAGENTA(i,j)>GraphDiameterMAGENTA
585               GraphDiameterMAGENTA=distMAGENTA(i,j);
              end
587         end
     end
589 end
```

A.2 Function prim_allocation

```
2 % Allocazione primitive circolari su imm_cluster1

4 function [Centri_1,Raggi_1,Aree] = prim_allocation(imm_cluster_1a)

6 CompConn_1a=bwconncomp(imm_cluster_1a,8);
Stats_1a=regionprops(CompConn_1a,'Area','Extrema','Eccentricity','
   Image');
8
imm_cluster_1aa=ismember(labelmatrix(CompConn_1a),find([Stats_1a.
   Area]>25));
10 CompConn_1aa=bwconncomp(imm_cluster_1aa,8);
Stats_1aa=regionprops(CompConn_1aa,'Area','Extrema','Eccentricity'
   ,'Image','PixelList');
12


14 imm_cluster_1c=ismember(labelmatrix(CompConn_1aa),find([Stats_1aa.
   Area]>=25));

16 % si applicano operatori morfologici per "ripulire" l'immagine:

18 BWer = imerode(imm_cluster_1c,strel('disk',3));
BWsp = bwmorph(BWer,'spur');
```

```
BWcl = bwmorph(BWsp,'clean');
BWfil = bwmorph(BWcl,'fill');
BWmag = bwmorph(BWfil,'majority');
Lab = bwlabel(BWmag);
Lab2 = Lab;

thrArea = (size(Lab,1)/100*size(Lab,2)/100)/10;
vec = zeros(1,max(Lab(:)));
for ctl=1:max(Lab(:))
    nn=find(Lab==ctl);
    vec(ctl)=length(nn);
    if vec(ctl)<thrArea
        Lab2(nn)=0;
    end
end
Lab2=bwlabel(Lab2);
imm_cluster_1c=Lab2>0;

CompConn_1c=bwconncomp(imm_cluster_1c,8);
Stats_1c=regionprops(CompConn_1c,'Area','Image','PixelIdxList','
   PixelList','Eccentricity');

NumPixelTot_1=size(imm_cluster_1c,1)*size(imm_cluster_1c,2);
Centri_1=zeros(NumPixelTot_1,2);
Raggi_1=zeros(NumPixelTot_1,1);
ContaCerchiValidi_1=1;

% scorro le componenti connesse del cluster 1: valuto cerchi
% centrati in ogni pixel e poi assegno ciascun pixel al
% cerchio maggiore che lo contiene:
%
% - valuto e "tengo" il cerchio dal più grande e annullo i
%   suoi pixel
% - valuto il secondo cerchio più grande, se non contiene
%    zeri, se non contiene zeri lo scarto, sennò lo "tengo"
% - e così via...
%

for i=1:CompConn_1c.NumObjects

    Immagine=(Stats_1c(i).Image)';


    Stats_1c(i).AreaCerchiProvvisori=zeros(Stats_1c(i).Area,1);
    Stats_1c(i).CentroCerchiProvvisori=zeros(Stats_1c(i).Area,2);
    Stats_1c(i).RaggiProvvisori=zeros(Stats_1c(i).Area,1);
    ContaCerchiProvvisori=1;

    for j=4:(size(Immagine,1)-3)
        for k=4:(size(Immagine,2)-3)

         if Immagine(j,k)==1

         w=1;

         while w==1
```

```
                    Raggio=0;

% controlli per non uscire fuori dall'immagine e per fermarsi al
    quadrato massimo che contiene solo 1

          while k-(Raggio+1)>0 && j-(Raggio+1)>0 &&k+Raggio+1<=size
              (Immagine,2) && j+Raggio+1<=size(Immagine,1)&& all(all
              (Immagine(j-(Raggio+1):j+Raggio+1,k-(Raggio+1):k+
              Raggio+1)))==1

% se si superano i controlli allora "allargo" il quadrato di un
    pixel
                    Raggio=Raggio+1;
           end

            if Raggio>=3

             % PixelList contiene solo gli 1
             index=nnz(Immagine(1:j-1,:))+nnz(Immagine(j,1:k));
             % index è l'indice nella matrice PixelList del pixel
                 unitario che ci interessa salvare come centro

             Stats_1c(i).CentroCerchiProvvisori(
                 ContaCerchiProvvisori,1)=Stats_1c(i).PixelList(
                 index,1);
             Stats_1c(i).CentroCerchiProvvisori(
                 ContaCerchiProvvisori,2)=Stats_1c(i).PixelList(
                 index,2);
             Stats_1c(i).AreaCerchiProvvisori(ContaCerchiProvvisori
                 )=pi*(Raggio)^2;
             Stats_1c(i).RaggiProvvisori(ContaCerchiProvvisori)=
                 Raggio;
             ContaCerchiProvvisori=ContaCerchiProvvisori+1;


              end
              w=0;
              end


             end


         end

    end
end
ImmagineBis=imm_cluster_1c';

for i=1:CompConn_1c.NumObjects

    while nnz(Stats_1c(i).AreaCerchiProvvisori)~=0

        % vedere paragrafo 2.2.2 per spiegazione dettagliata
        [~,Indice]=max(Stats_1c(i).AreaCerchiProvvisori);

```

```
        j=Stats_1c(i).CentroCerchiProvvisori(Indice,1);
120     k=Stats_1c(i).CentroCerchiProvvisori(Indice,2);
        Raggio=Stats_1c(i).RaggiProvvisori(Indice);
122
        % Controllo che il quadrato centrato in (j,k) e di raggio
            Raggio sia costituito di soli 1: sommo tutti gli
            elementi della
124     % sottomatrice e verifico che tale somma sia uguale al
            numero di elementi della sottomatrice, cioè [(Raggio*2)
            +1]^2.
        %
126     % > se la risposta è affermativa salvo il centro del
            cerchio, il raggio e l'area

128         if sum(sum(ImmagineBis(j-Raggio:j+Raggio,k-Raggio:k+
                Raggio)))==((2*Raggio)+1)^2

130          Centri_1(ContaCerchiValidi_1,1)=Stats_1c(i).
                CentroCerchiProvvisori(Indice,1);
             Centri_1(ContaCerchiValidi_1,2)=Stats_1c(i).
                CentroCerchiProvvisori(Indice,2);
132          Raggi_1(ContaCerchiValidi_1)=Stats_1c(i).
                RaggiProvvisori(Indice);
             ImmagineBis(j-Raggi_1(ContaCerchiValidi_1):j+Raggi_1(
                ContaCerchiValidi_1),k-Raggi_1(ContaCerchiValidi_1)
                :k+Raggi_1(ContaCerchiValidi_1))=zeros(2*(Raggi_1(
                ContaCerchiValidi_1))+1);
134
             Stats_1c(i).AreaCerchiProvvisori(Indice)=0;
136          Stats_1c(i).CentroCerchiProvvisori(Indice,:)=[0 0];
             Stats_1c(i).RaggiProvvisori(Indice)=0;
138
             ContaCerchiValidi_1=ContaCerchiValidi_1+1;
140
           % > se la risposta è falsa azzero tutte le caselle dei
              vettori provvisori corrispondenti a tale cerchio in
              modo da non tenerne
142        % più conto nelle valutazioni successive
           else
144          Stats_1c(i).AreaCerchiProvvisori(Indice)=0;
             Stats_1c(i).CentroCerchiProvvisori(Indice,:)=[0 0];
146          Stats_1c(i).RaggiProvvisori(Indice)=0;
            end
148


150
    end
152 end

154 Aree = pi*(Raggi_1).^2;
```

Bibliografia

[1] D. Altunbay, C.Cigir, C.Sokmensuer, C.Gunduz-Demir, *Color Graphs for Automated Cancer Diagnosis and Grading*, IEEE Transactions on Biomedical Engeneering, vol.57,no.3, 2010.

[2] A. B. Tosun, M. Kandemir, C.Sokmensuer, C.Gunduz-Demir, *Object-oriented texture analysis for the unsupervised segmentation of biopsy images for cancer detection*, Pattern recognition, vol.42, no.6, 2009.

[3] W. N. Street, W. H. Wolberg, and O. L. Mangasarian, *Nuclear feature extraction for breast tumor diagnosis*, International Symposium on Electronic Imaging: Science and Tecnology,vol.1905, pp. 861-870, 1993.

[4] M. Wiltgen, A. Gerger, J. Smolle, *Tissue counter analysis of benign common nevi and malignant melanoma*, International Journal of medical informatics, 2007.

[5] S. Doyle, M. Hwang, K. Shah, A. Madabhushi, M. Feldman, J.Tomaszewski, *Automated grading of prostate cancer using architectural and textural image features*, Biomedical Imaging From Nano to Macro, pp 1284-1287.

[6] A. Esgiar, R. Naguib, B. Sharif, M. Bennett A. Murray, *Microscopic image analysis for quantitative measurement and feature identification of normal and cancerous colonic mucosa*, IEEE Transactions on Information Technology in Biomedicine, vol.2, no.3, 1998.

[7] C. Demir, S. Humayun Gultekin, Bulent Yener, *Learning the Topological Properties of Brain Tumors*, IEEE Transactions on computational Biology and Bioinformatics, vol.2, no.3, 2005.

[8] S.Theodoridis, K. Koutroumbas, *Pattern Recognition*, Elsevier, 2009.

[9] G. Bianchi Porro, F. Pace, *Argomenti di patologia esofagea*, Springer, 2001.