Alma Mater Studiorum – Università di Bologna – Campus di Cesena

Dipartimento Informatica – Scienza e Ingegneria (DISI)

Corso di Laurea in Ingegneria e Scienze Informatiche

Analisi di macrofagi e linfociti in immagini di provini istologici di carcinoma a cellule squamose del distretto testa-collo attraverso scripts sviluppati per il software open-source *QuPath*.

Tesi in: Programmazione

Relatore

Prof.ssa Antonella Carbonaro

Presentata da

Ambrogiani Alessandro

Correlatori

Prof. Filippo Piccinini

Dott.ssa Maria Maddalena Tumedei

Dott.ssa Rebecca Polidori

Dott.ssa Marcella Tazzari

Anno Accademico 2024 / 2025

"*Il successo non è mai definitivo, il fallimento non è mai fatale; è il coraggio di continuare che conta.*"

Sir Winston Churchill

Parole Chiave

Istologia

Microscopia

Analisi a singola cellula

Open-source software

QuPath

Abstract (English version)

This thesis mainly focuses on the quantitative analysis of lymphocytes and macrophages in images of Head and Neck Squamous Cell Carcinoma using the open-source software *QuPath* supported by custom scripts. The project lies within the field of digital image processing applied to histological data, aiming to automate and standardize cell identification procedures in neoplastic tissues.

After studying the morphological characteristics of the main cell types and their corresponding immunohistochemical markers, custom scripts were developed and tested mainly in the *Groovy* language to optimize cell segmentation and classification. The automation process reduced manual intervention and improved reproducibility compared to traditional analysis methods.

Experimental results demonstrated good accuracy in distinguishing lymphocytes from macropahges in Head and Neck Squamous Cell Carcinoma tumor specimens, suggesting potential applications in diagnostic workflows and translational research. This work highlights the effectiveness of integrating computer science and biomedical expertise for the automated management and interpretation of complex histopathological images.

Abstract (versione italiana)

Il presente lavoro di Tesi ha come obiettivo principale l'analisi quantitativa di linfociti e macrofagi in immagini di carcinoma a cellule squamose del distretto testa-collo, utilizzando il software open-source *QuPath* supportato da script customizzati. L'attività si inserisce nel contesto dell'elaborazione digitale di immagini istologiche, con l'intento di automatizzare e standardizzare i processi di identificazione cellulare in tessuti neoplastici.

Dopo una fase di studio delle caratteristiche morfologiche dei diversi tipi cellulari e dei marker immunoistochimici associati, sono stati sviluppati e testati script personalizzati principalmente in linguaggio Groovy per ottimizzare la segmentazione e la classificazione delle cellule. L'automatizzazione ha consentito di ridurre l'intervento manuale e di migliorare la riproducibilità delle analisi rispetto ai metodi tradizionali.

I risultati sperimentali mostrano una buona accuratezza nella distinzione tra linfociti e macrofagi in provini di carcinoma a cellule squamose del distretto testa-collo, aprendo prospettive di applicazione in ambito diagnostico e di ricerca traslazionale. Il lavoro dimostra l'efficacia dell'integrazione tra competenze informatiche e biomediche nella gestione e interpretazione automatizzata di immagini istopatologiche complesse.

Indice

Introduzione

Il presente progetto di Tesi si inserisce nelle attività del gruppo di ricerca "Data Science for Health" (DS4H), una collaborazione multidisciplinare tra l'Università di Bologna (UniBo) e l'Istituto Romagnolo per lo Studio dei Tumori "Dino Amadori" IRCCS (IRST) di Meldola (FC), che integra competenze informatiche, ingegneristiche, fisiche e biomediche con l'obiettivo di sviluppare approcci quantitativi e metodologie computazionali a supporto della ricerca clinica. Lo studio è focalizzato sul carcinoma a cellule squamose del distretto testa-collo, una neoplasia solida caratterizzata da elevata eterogeneità biologica e clinica, spesso associata a una prognosi sfavorevole nelle forme localmente avanzate o recidivanti. Nonostante i progressi terapeutici ottenuti con chirurgia, radioterapia, chemioterapia e più recentemente con l'immunoterapia, una quota significativa di pazienti sviluppa recidiva di malattia o resistenza ai trattamenti. La complessità del microambiente tumorale e l'assenza di biomarcatori prognostici e predittivi robusti rendono necessario lo sviluppo di strumenti più accurati per la stratificazione dei pazienti e l'ottimizzazione delle strategie terapeutiche[1].

L'obiettivo del progetto è sviluppare un approccio di patologia digitale per l'identificazione di potenziali marcatori prognostici basati sull'analisi quantitativa del microambiente tumorale. In particolare, il lavoro si concentra sulla caratterizzazione geometrica e spaziale di macrofagi in immagini istologiche di pazienti affetti da carcinoma a cellule squamose del distretto testa-collo, valutando la loro distribuzione e la distanza rispetto alle strutture linfoidi organizzate, note in letteratura come "Tertiary Lymphoid Structures" (TLS). Le TLS sono aggregati organizzati di cellule immunitarie che si formano ectopicamente all'interno o in prossimità del tessuto tumorale. Queste strutture mimano l'architettura dei linfonodi e sono coinvolte nella regolazione della risposta immunitaria antitumorale. In base al grado di organizzazione, le TLS possono essere distinte in forme immature e mature: queste ultime sono caratterizzate dalla presenza di una rete follicolare $CD21^{+}$, indicativa di centri germinativi funzionali[2].

Il flusso di lavoro inizia dalla preparazione dei provini istologici mediante protocolli standard di colorazione istopatologica e immunoistochimica, a partire da campioni reali di pazienti forniti dalla Dott.ssa Marcella Tazzari e dalla Dott.ssa Maria Maddalena Tumedei, nell'ambito delle attività clinico-assistenziali dell'IRST. I vetrini vengono digitalizzati ad alta risoluzione tramite lo scanner

“Aperio CS2 Leica”, ottenendo immagini di tipo “Whole Slide Image” (WSI) che permettono la valutazione morfologica dell’intera sezione tissutale. Su queste immagini vengono annotate le regioni di interesse (ROI, Region Of Interest) nelle quali è condotto lo studio geometrico di linfociti e macrofagi.

Le ROI selezionate per ciascun provino istologico vengono quindi esportate in formato “.tiff” tramite il software *Aperio ImageScope*, costituendo il punto di partenza per la fase di pre-processing delle immagini. In questa fase è stato sviluppato uno script dedicato per *Fiji/ImageJ*, che converte l’immagine “.tiff” della ROI in una rappresentazione in pseudo-fluorescenza in formato “.png”, al fine di migliorare il contrasto tra le componenti cellulari di interesse e facilitare le successive operazioni di segmentazione e riconoscimento automatico.

La fase centrale del progetto prevede l’impiego di *QuPath*, software open source progettato per l’analisi e la visualizzazione di immagini biomediche e, in particolare, di vetrini istologici digitali ad alta risoluzione. All’interno di *QuPath* le immagini pseudo-fluorescenti vengono ulteriormente annotate identificando TLS mature e immature e l’area di studio viene suddivisa in sotto-regioni (tiles), in modo da ottenere una descrizione più granulare dell’eterogeneità spaziale del tessuto. Su ciascun tile viene eseguita una serie di script custom, creati sotto la supervisione dalla Dott.ssa Rebecca Polidori (IRCCS Istituto Clinico Humanitas Research Hospital, Milano), il cui scopo è la rilevazione automatica delle cellule (cell detection) e l’assegnazione di misure geometriche quali area, perimetro, diametro equivalente, circularity e altre feature morfometriche, oltre alla quantificazione e la distribuzione dei macrofagi in relazione alle TLS annotate.

I risultati di questa fase vengono esportati in formato “.csv”, ottenendo per ogni cellula un insieme strutturato di misurazioni che descrivono sia le sue caratteristiche geometriche sia la sua posizione relativa rispetto alle TLS e ad altre strutture tissutali. Questo approccio consente di trasformare l’informazione visiva contenuta nelle immagini istologiche in dati quantitativi utilizzabili per successive analisi statistiche e modelli predittivi.

Nella fase finale del progetto, i file “.csv” generati da *QuPath* vengono importati in *RStudio* per l’analisi statistica, utilizzando script dedicati per lo studio delle relazioni tra le diverse popolazioni

cellulari e le TLS. Per ogni tile vengono calcolate misure riassuntive, tra cui la distanza media delle cellule di specifici tipi (ad esempio macrofagi e linfociti) dalle TLS, con l'obiettivo di individuare pattern spaziali potenzialmente associati a differenti profili clinici. Queste analisi costituiscono la base per future valutazioni su coorti più ampie, volte a validare l'uso di descrittori geometrici e spaziali del microambiente tumorale come marcatori prognostici nel linfoma follicolare.

La proposta di questo progetto di Tesi è stata formulata dalla Prof.ssa Antonella Carbonaro (UniBo) e dal Prof. Filippo Piccinini (UniBo, IRST), nell'ambito delle attività congiunte del gruppo DS4H, utilizzando come caso di studio immagini istologiche di pazienti fornite dalla Dott.ssa Marcella Tazzari (IRST) e dalla Dott.ssa Maria Maddalena Tumedei (IRST), assicurando così una forte integrazione tra ricerca metodologica, pratica clinica e bisogni reali dei pazienti.

1. Analisi di linfociti e macrofagi in provini istologici

L'istologia (dal greco histos, "tessuto", e logos, "studio") è la branca della biologia che studia la struttura microscopica dei tessuti animali e vegetali. Nel contesto oncologico, l'analisi istologica rappresenta il "Gold Standard" per la diagnosi tumorale, permettendo di valutare l'architettura tissutale e le alterazioni morfologiche cellulari che distinguono un tessuto sano da uno neoplastico [3].

Nel presente studio è stata condotta un'analisi quantitativa e spaziale delle popolazioni cellulari nel microambiente tumorale. Sono stati esaminati i seguenti marcatori di superficie per identificare specifiche sottopopolazioni:

- **CD163 (macrofagi):** Questo marcatore identifica una popolazione di macrofagi associati al tumore (TAM, Tumor-Associated Macrophages). Nel carcinoma a cellule squamose del distretto testa-collo, queste cellule immunitarie innate, pur possedendo capacità fagocitiche e di presentazione dell'antigene, possono essere "educate" dalle cellule neoplastiche. Tale cooptazione le porta a esercitare un ruolo pro-tumorale, favorendo la crescita, la sopravvivenza cellulare e l'immunosoppressione.
- **CD3 (linfociti T totali):** Questo complesso proteico è espresso costitutivamente sulla superficie di tutti i linfociti T ed è fondamentale per l'attivazione del segnale intracellulare del recettore delle cellule T (TCR). Nel contesto dello studio, il CD3 è stato utilizzato come marcatore pan-T per quantificare l'infiltrato linfocitario globale (TILs, Tumor-Infiltrating Lymphocytes), che comprende sia le sottopopolazioni helper (CD4) che citotossiche (CD8), fornendo una stima della risposta immunitaria adattativa complessiva presente nel tessuto.
- **CD8 (linfociti T citotossici):** Questo marcatore identifica i linfociti T, effettori principali dell'immunità cellulo-mediata. La loro funzione è riconoscere ed eliminare specificamente le cellule infettate o trasformate (malate), rappresentando quindi un attore cruciale della risposta immunitaria anti-tumorale.
- **CD20 (linfociti B):** È un antigene transmembrana specifico della cellule B, espresso dallo stadio pre-B fino alla differenziazione in cellule B della memoria (ma assente nelle plasmacellule).

Preparazione di un provino istologico

Il processo di preparazione del campione istologico permette di processare un campione di tessuto biologico in modo che possa essere osservato al microscopio e studiato nei suoi dettagli cellulari. Le fasi fondamentali per allestire un preparato istologico sono:

1. Prelievo
2. Fissazione
3. Inclusione
4. Sezionamento
5. Colorazione
6. Montaggio

Prelievo del campione

Consiste nel raccogliere il campione tissutale. La preparazione del paziente e la tecnica di prelievo variano a seconda della sede e del tipo di biopsia.

Fissazione

La fissazione rappresenta la fase preliminare e più critica dell'intero processo istopatologico, definita come l'utilizzo di specifici agenti chimici per preservare i costituenti tissutali e cellulari in uno stato quanto più possibile prossimo alla condizione in vivo. L'obiettivo primario di tale procedura è l'arresto immediato dei processi degenerativi post-mortem, prevenendo sia l'autolisi, ovvero la degradazione enzimatica operata dagli organelli cellulari stessi, sia la putrefazione causata dalla proliferazione di agenti batterici.

Oltre alla stabilizzazione biologica, il processo mira a conferire al campione una consistenza fisica adeguata, indurendolo per proteggerlo dai traumi meccanici derivanti dalle successive fasi di processamento, come la disidratazione e il taglio al microtomo. Contemporaneamente, l'azione del fissativo stabilizza la morfologia cellulare prevenendo distorsioni osmotiche e modifica le proprietà

chimiche delle proteine, aumentando significativamente l'affinità del tessuto per i coloranti istologici e per i reagenti impiegati nelle tecniche immunoistochimiche .

Dal punto di vista biochimico, i fissativi operano principalmente attraverso due meccanismi d'azione: la coagulazione e la formazione di legami incrociati. I fissativi coagulanti, come l'etanolo, agiscono rimuovendo le molecole d'acqua e inducendo la precipitazione delle proteine; al contrario, agenti come la formaldeide stabilizzano l'architettura tissutale creando ponti chimici, definiti cross-links, tra le catene proteiche, bloccandole in un reticolo rigido che ne preserva la struttura spaziale. L'adeguatezza di questa fase è determinante per l'intero percorso diagnostico, poiché una fissazione inadeguata compromette irreversibilmente tutte le operazioni successive, rendendo impossibile una valutazione patologica accurata [4].

Disidratazione e Diafanizzazione

Terminata la fissazione, il campione deve essere disidratato, ovvero tutta l'acqua contenuta nelle cellule deve essere progressivamente allontanata e sostituita con alcol assoluto, al fine di consentire la successiva fase di inclusione. Una volta completata la disidratazione, è necessario diafanizzare il campione, ovvero immergerlo in una sostanza chimica miscibile sia con gli alcoli sia con il mezzo di inclusione, per renderlo trasparente, facilitando l'attraversamento da parte del fascio luminoso del microscopio.

Inclusione

Avendo rimosso tutta l'acqua dalle cellule del campione e avendola sostituta totalmente con il diafanizzante, è possibile procedere alla fase dell'inclusione, il cui scopo è quello di indurire il campione e soprattutto di includerlo all'interno di un supporto/mezzo rigido che potrà essere, poi, tagliato per ottenere delle sezioni estremamente sottili.

Taglio (Sezionamento)

La fase di taglio prevede l'utilizzo di uno strumento di laboratorio chiamato microtomo, mediante il quale si ricavano sezioni molto sottili, di circa 3-4 micron, in modo da consentire il passaggio della luce e garantire che, dopo la colorazione, le strutture cellulari siano chiaramente osservabili al microscopio.

Colorazione

Le cellule che sono normalmente trasparenti dato che sono costituite essenzialmente da acqua e, pertanto, devono essere colorate artificialmente per poter essere osservate al microscopio. Esistono 2 tipi principali di colorazioni:

- **Colorazioni fisiche:** Vengono utilizzati coloranti che si distribuiscono in specifiche strutture del campione senza instaurare legami chimici con queste ultime.
- **Colorazioni chimiche:** Utilizzano coloranti che reagiscono chimicamente con componenti specifiche del tessuto, formando legami stabili che ne permettono l'evidenziazione.

Tra le colorazioni chimiche si colloca l'immunoistochimica, che utilizza anticorpi che si legano selettivamente agli antigeni presenti nelle cellule, permettendo di localizzare con precisione proteine o altri marcatori biologici. Questo approccio fornisce informazioni dettagliate non solo sulla morfologia, ma anche sulla distribuzione di specifiche molecole, risultando fondamentale sia per studi di ricerca sia per applicazioni diagnostiche[5].

Nel presente lavoro, le analisi sono state eseguite utilizzando l'immunocoloratore Benchmark ULTRA **(**Roche**)**, che consente di eseguire doppie colorazioni immunoistochimiche, permettendo di evidenziare simultaneamente due marcatori diversi nello stesso campione e aumentando così la quantità di informazioni ottenibili da ogni sezione.

Montaggio

L’ultima fase della preparazione del campione istologico è il montaggio, durante il quale il campione processato viene coperto con un mezzo di montaggio e poi con un vetrino copriogetto. Il mezzo di montaggio solidifica, rendendo il campione stabile e permettendo la sua osservazione al microscopio.

2. Introduzione a QuPath per analisi di immagini istologiche

QuPath è un software open source di bioimage analysis progettato per soddisfare la crescente domanda di strumenti user-friendly, estendibili ed efficienti nell'ambito della patologia digitale e dell'analisi delle Whole Slide Images (WSI). L'acquisizione di immagini digitali ad altissima risoluzione di interi vetrini istologici sta rivoluzionando il settore biomedico, rendendo necessario lo sviluppo di soluzioni in grado di gestire e analizzare grandi insiemi di dati con algoritmi scalabili e risultati riproducibili.

Rispetto ad altri software open source, *QuPath* si distingue per la sua architettura ottimizzata specificamente per le WSI. Integra un visualizzatore multi-thread e tile-based che permette una navigazione fluida ed efficiente anche su immagini di decine di gigabyte. Il vero punto di forza tecnologico di *QuPath* è il suo modello dati gerarchico "object-based": ogni oggetto rilevato nell'immagine (nucleo, cellula, regione annotata, ecc.) viene rappresentato come entità relazionale, consentendo analisi complesse e la costruzione di workflow sia tramite operazioni interattive, sia mediante script personalizzati. Le principali funzionalità di *QuPath* includono:

- Strumenti di annotazione manuale e automatizzata;
- Segmentazione di nuclei e cellule;
- Estrazione di feature morfologiche e di intensità;
- Classificazione supervisionata degli oggetti mediante algoritmi di machine learning (come *Random Forest*);
- Ampio set di algoritmi predefiniti e la possibilità di estendere la piattaforma con nuove soluzioni condivisibili dalla comunità scientifica.

QuPath permette di elaborare interi progetti in modalità batch, consente analisi ad alta produttività su ampie coorti di campioni e fornisce l'esportazione automatica dei risultati per successiva integrazione statistica. È stato ampiamente validato su casi reali, dimostrando rapidità e accuratezza, per esempio nel conteggio di marcatori immunoistochimici (CD3, CD8, p53, PD-L1) su migliaia di "tissue microarray" (TMA) e nell'analisi quantitativa del microambiente tumorale.

Grazie agli strumenti integrati, è possibile associare i dati di imaging ad endpoint clinici come sopravvivenza e risposta ai trattamenti, ampliando l'impatto della patologia digitale nella ricerca traslazionale e nell'attività diagnostica.

2.1. Modello gerarchico ad oggetti

La vera innovazione tecnica introdotta da *QuPath* risiede nella struttura dati gerarchica basata su oggetti, pensata specificatamente per la gestione di immagini a elevata complessità e dimensione come le WSI. In *QuPath*, ogni "oggetto" rappresenta qualsiasi entità logica riconoscibile nell'immagine digitale. Un oggetto può essere ad esempio:

- una annotazione realizzata manualmente dall'utente (come una regione di interesse, ROI),
- oppure una struttura identificata automaticamente dal software, come un nucleo cellulare, una cellula intera, un'area di tessuto o una regione tumorale.

La gestione gerarchica significa che gli oggetti sono organizzati in una struttura "padre-figlio". Ad esempio, una ROI disegnata manualmente può diventare il "contenitore" di centinaia o migliaia di oggetti "cellula", ciascuno segmentato in modo automatico nell'area selezionata. Questa organizzazione permette di:

- Raggruppare oggetti all'interno di regioni definite.
- Mantenere precise relazioni logiche fra le differenti scale di dettaglio (dall'intera immagine fino al singolo nucleo).
- Assegnare ad ogni oggetto attributi specifici, come la classificazione morfologica ("tumorale", "linfocita", "macrofafo", ecc.) e le misurazioni quantitative (ad esempio, area, intensità del marcatore immunoistochimico, perimetro, circolarità, mean optical density).

Il modello gerarchico di *QuPath* consente quindi la gestione efficiente di milioni di oggetti, aspetto fondamentale per immagini gigapixel, e offre la possibilità di addestrare rapidamente classificatori

di machine learning sfruttando feature associate a ogni oggetto, senza dover esportare i dati verso software esterni.

2.2. Estensibilità e Interoperabilità

A differenza di molti altri software per la bioimage analysis, *QuPath* è stato concepito fin dall'inizio per essere altamente modulare e facilmente estendibile. Questo significa che, oltre alle funzionalità base incluse nel programma, gli utenti e gli sviluppatori possono:

- Creare e integrare nuove estensioni e plugin, aggiungendo funzioni innovative senza dover modificare il codice sorgente principale.
- Sfruttare un'API documentata e progettata per facilitare lo sviluppo e lo scambio di moduli aggiuntivi nelle comunità scientifiche di riferimento.

Dal punto di vista dell'interoperabilità, *QuPath* supporta il trasferimento di dati e misurazioni verso software esterni ampiamente utilizzati come *ImageJ*, *MATLAB* o programmi di analisi statistica quali *R*. Questo scambio è particolarmente importante perché consente di sfruttare la potenza dei diversi ambienti software (ad esempio per ulteriori analisi di tipo quantitativo o statistico) mantenendo al contempo le potenzialità di esplorazione e segmentazione offerte da *QuPath*.

L'estensibilità e l'integrazione con altri strumenti rendono *QuPath* una scelta ideale non solo per la ricerca accademica ma anche per laboratori clinici che necessitano di piattaforme solide, aggiornabili e in grado di evolvere con le esigenze della comunità scientifica.

2.3. Piattaforma di analisi flessibile

QuPath non si limita a essere un semplice visualizzatore di immagini, ma è una piattaforma completa e cross-platform per l'analisi avanzata delle WSI. Il software è dotato di un set di strumenti "pronti all'uso" (out-of-the-box) che rispondono ai bisogni pratici della patologia digitale:

- Algoritmi per la segmentazione automatica di nuclei e cellule, con parametri adattabili a seconda del tessuto o della colorazione utilizzata.
- Strumenti dedicati alla quantificazione di biomarcatori, fondamentali per l'analisi immunoistochimica in ricerca e in clinica.
- Funzionalità di visualizzazione efficace, che consentono zoom dinamico, gestione di livelli multipli di ingrandimento e visualizzazione di molteplici vetrini in parallelo.

Un punto chiave della flessibilità di *QuPath* sta nella possibilità di automatizzare interi workflow mediante scripting (principalmente con il linguaggio *Groovy*): l'utente può scrivere o adattare script per ripetere analisi standardizzate su interi dataset, esportare i risultati in modo strutturato (ad esempio, in formato CSV), eseguire batch processing su decine o centinaia di immagini in sequenza e produrre report automatizzati. Questa capacità di personalizzare i flussi di lavoro si traduce in maggiore efficienza, riproducibilità e adattabilità per obiettivi specifici, dalla ricerca di base alla routine clinica [6].

3. Script Fiji/ImageJ per creazione immagini pseudo-fluorescenza

Prima di analizzare nel dettaglio lo script utilizzato, è opportuno introdurre la piattaforma software su cui è stato sviluppato.

3.1. Fiji/ImageJ

Fiji (acronimo ricorsivo per *Fiji Is Just ImageJ*) rappresenta una distribuzione avanzata del software *ImageJ*, progettata specificamente per affrontare le sfide della moderna analisi di bioimmagini.

Questa piattaforma supera i limiti della versione standard attraverso un'architettura definita 'batteries-included', che integra nativamente librerie essenziali come *Bio-Formats*, per la gestione di formati microscopici proprietari, e *ImgLib2*, per l'elaborazione agnostica di dati multidimensionali. L'obiettivo primario dell'ambiente *Fiji* è colmare il divario tecnologico tra lo sviluppo di algoritmi computazionali complessi e la loro applicazione pratica nelle scienze della vita.

Un aspetto centrale della piattaforma, di particolare rilevanza per questo progetto di tesi, risiede nella sua potente infrastruttura di scripting. *Fiji* supporta infatti la prototipazione rapida di flussi di lavoro personalizzati, permettendo l'automazione di compiti ripetitivi e garantendo la riproducibilità scientifica dei risultati; tale caratteristica si è rivelata l'elemento chiave che ha permesso lo sviluppo e l'implementazione della macro presentata in questo studio [7].

3.2. Script pseudo-fluorescenza

Per passare dall'immagine istologica in campo chiaro (RGB) a un'immagine a pseudo-fluorescenza quantificabile, è stata sviluppata una macro personalizzata in linguaggio *ImageJ Macro Language* (IJL) denominata "MACRO_2_CD20CD3".

Questa macro guida l'utente attraverso un flusso di lavoro semi-automatico progettato per isolare i segnali dei biomarcatori, segmentarli e ripulirli da artefatti, garantendo al contempo la riproducibilità attraverso un log dettagliato.

Thresholding

Il thresholding (in italiano "sogliare" o "binariazione") è una delle tecniche fondamentali nell'analisi delle immagini digitali, soprattutto in ambito biomedicale. Consiste nel trasformare un'immagine in una versione semplificata, spesso binaria, dove i pixel vengono suddivisi in due (o più) gruppi in base al loro valore di intensità o colore.

Come funziona

Il processo di thresholding parte dall'analisi dell'istogramma dei valori di intensità dei pixel (ad esempio, da una scala di grigi o singolo canale di colore). Si sceglie quindi una soglia (threshold):

- I pixel con valori superiori alla soglia vengono classificati in un gruppo (ad esempio, "positivi" o "foreground").
- I pixel inferiori finiscono nell'altro gruppo ("negativi" o "background").

Questo metodo semplice è spesso usato in istologia per distinguere cellule marcate ("positive" per un certo segnale, come la DAB nell'immunoistochimica) dal fondo del tessuto.

3.2.1. Inizializzazione e Impostazione (Setup)

La macro inizia chiudendo tutte le finestre aperte per evitare errori. Successivamente, guida l'utente nella configurazione dell'analisi:

1. **Input:** Richiede la selezione del file immagine da elaborare.
2. **Parametri:** Chiede all'utente di definire un nome o un identificativo specifico per ROI analizzata.
3. **Output:** Richiede la selezione di una directory di output. All'interno di questa, crea una sotto-cartella univoca utilizzando il nome dell'immagine e della ROI, per organizzare tutti i file generati.
4. **Log:** Inizializza un file di log all'interno della cartella di output, dove verranno registrate tutte le scelte e i parametri dell'analisi.

Codice sviluppato:

```
//Define input file
run("Close All");
path1 = File.openDialog("Select a File");
dir1 = File.getParent(path1);
name1 = File.getName(path1);
print("Path:", path1);
print("Name:", name1);
print("Directory:", dir1);
dotIndex = lastIndexOf(name1, ".");
if (dotIndex != -1) {
    name1WithoutExtension = substring(name1, 0, dotIndex);
} else {
    name1WithoutExtension = name1;
}

//Define output parameters
outputROIname1 = getString("Enter the name for the ROI analysed: ", "");
dir2 = getDirectory("Set the output directory (subfolders will be automatically
created): ");
dir2Subfolder = dir2 + "/" + name1WithoutExtension + "_" + outputROIname1 + "/";
```

```
File.makeDirectory(dir2Subfolder);

// Create log file
logFile = dir2Subfolder + name1WithoutExtension + outputROIname1 +
"_processing_log.txt";
```

3.2.2. Deconvoluzione Spettrale (Da RGB a Pseudo-Fluorescenza)

Questo è il passaggio scientifico centrale della conversione:

1. **Apertura e controllo:** L'immagine viene aperta e lo script ne controlla la profondità di bit. Se non è già un'immagine RGB a 24 bit, viene convertita.
2. **Separazione dei coloranti**: Viene eseguito il comando *Colour Deconvolution* con i vettori predefiniti "FastRed FastBlue DAB". Questa funzione separa l'immagine RGB originale in tre canali distinti, ciascuno rappresentante l'intensità di uno specifico colorante istologico.
3. **Normalizzazione dei canali:** Per ciascuno dei tre canali separati, la macro apre un box di dialogo che consente all'utente di associare il canale al marcatore biologico corretto (es. Canale 1 = "CD3", Canale 2 = "NUCLEI", Canale 3 = "CD20").
4. **Inversione e salvataggio:** Ciascun canale viene convertito a 8 bit, invertito (per trasformare il segnale scuro del colorante in un segnale luminoso, simile alla fluorescenza) e salvato come file TIFF _BN.tif. Viene inoltre creato un duplicato _BN-1.tif che servirà per la segmentazione.

Codice sviluppato:

```
//Open input file
list = getFileList(dir1);
print("Directory contains "+list.length+" files");
open(path1);
selectImage(nImages);
bitDepth1 = bitDepth();
print(bitDepth1);

if(bitDepth1 == 24){
    print("Image is RGB.");
} else {
```

```
    print("Image is not RGB.");
    run("Stack to RGB");
    selectImage(nImages);
    close();
    selectImage(nImages);
    rename(name1);
}

if (bitDepth1 != 24) {
    run("RGB Color");
}

// Save ColourDeconvolution images
run("Colour Deconvolution", "vectors=[FastRed FastBlue DAB]");

// CH1 -> CD3 (Yellow)
selectImage(name1+" (RGB)-(Colour_1)");
Dialog.create("");
Dialog.addMessage("Select name for CH1");
Dialog.addChoice("name", newArray("CD3"), "CD3");
Dialog.show();
outputCH1name1 = Dialog.getChoice();
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + ".tif");
run("8-bit");
run("Invert");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_BN.tif");
run("Duplicate...", " ");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_BN-1.tif");

// CH2 -> NUCLEI (Blue)
selectImage(name1+" (RGB)-(Colour_2)");
Dialog.create("");
Dialog.addMessage("Select name for CH2");
Dialog.addChoice("name", newArray("NUCLEI"), "NUCLEI");
Dialog.show();
outputCH2name1 = Dialog.getChoice();
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + ".tif");
run("8-bit");
run("Invert");
```

```
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_BN.tif");
run("Duplicate...", " ");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_BN-1.tif");

// CH3 -> CD20 (Magenta)
selectImage(name1+" (RGB)-(Colour_3)");
Dialog.create("");
Dialog.addMessage("Select name for CH3");
Dialog.addChoice("name", newArray("CD20"), "CD20");
Dialog.show();
outputCH3name1 = Dialog.getChoice();
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + ".tif");
run("8-bit");
run("Invert");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_BN.tif");
run("Duplicate...", " ");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_BN-1.tif");

run("Close All");
```

3.2.3. Segmentazione e Creazione Maschere (Processo Semi-Automatico)

Questa fase, gestita dalla funzione “processChannel”, è finalizzata alla generazione di una maschera binaria per l'isolamento delle aree di segnale "positivo" specifiche per ciascun canale. L'algoritmo implementa una logica iterativa (loop): il sistema ripropone ciclicamente l'elaborazione, consentendo all'operatore di affinare i parametri in tempo reale fino all'esplicita validazione del risultato finale. Il processo è suddiviso nelle seguenti fasi:

1. **Thresholding (sogliatura):** Per ogni canale (CD3, NUCLEI, CD20), l'utente sceglie un algoritmo di thresholding automatico (es. "Otsu", "Yen", "Moments"). Questo algoritmo analizza l'istogramma dell'immagine e determina la soglia ottimale per separare il segnale (foreground) dallo sfondo (background).

2. **Filtraggio (opzionale):** L'utente può scegliere di applicare un filtro mediano per ridurre il rumore (pixel spuri).
3. **Creazione maschera:** L'immagine viene convertita in una maschera binaria _TH.tif basata sulla soglia e sul filtro scelti.

Codice sviluppato:

```
// FUNCTION: PROCESS SINGLE CHANNEL WITH PREVIEW
function processChannel(channelName, shortName, defaultAlgo, defaultRadius) {

    originalImage = dir2Subfolder + name1WithoutExtension + outputROIname1 + "_"
+ channelName + "_BN-1.tif";

    // Variables to store final choices
    finalAlgo = defaultAlgo;
    finalMedianSelection = "NO";
    finalRadius = defaultRadius;

    // PREVIEW LOOP
    while(true) {
        run("Close All");
        open(originalImage);
        originalTitle = getTitle();
        run("Duplicate...", "title=PreviewImage");
        selectImage("PreviewImage");

        // 1. Ask parameters
        Dialog.create("Settings for " + shortName);
        Dialog.addMessage("Adjust settings. A preview will be shown next.");
        Dialog.addChoice("Threshold Algorithm:",
             newArray("Default", "Huang", "Otsu", "Minimum", "Triangle", "Mean",
"Moments", "Percentile", "Yen", "Intermodes"),
            finalAlgo);

        medianCheck = false;
        if (finalMedianSelection == "YES") { medianCheck = true; }

        Dialog.addCheckbox("Apply Median Filter", medianCheck);
        Dialog.addNumber("Median Radius:", finalRadius);
        Dialog.show();
```

```
        currentAlgo = Dialog.getChoice();
        isMedian = Dialog.getCheckbox();
        currentRadius = Dialog.getNumber();

        // 2. Apply Process on Preview Image
        selectImage("PreviewImage");

        if (isMedian) {
            run("Median...", "radius=" + currentRadius);
        }

        setAutoThreshold(currentAlgo + " dark no-reset");
        setOption("BlackBackground", true);
        run("Convert to Mask");

        // 3. Show Result and Ask for Confirmation
        selectImage("PreviewImage");
        setLocation(100, 100);

        medianText = "NO";
        if (isMedian) { medianText = "YES"; }

        Dialog.create("Preview Result: " + shortName);
        Dialog.addMessage("Check the 'PreviewImage'.\nAlgorithm: " + currentAlgo
+ "\nMedian: " + medianText + " (Radius: " + currentRadius + ")");
             Dialog.addChoice("Are you satisfied?", newArray("YES", "NO - Try
Again"), "YES");
        Dialog.show();

        satisfaction = Dialog.getChoice();

        if (satisfaction == "YES") {
            finalAlgo = currentAlgo;
            finalRadius = currentRadius;
                        if (isMedian) { finalMedianSelection = "YES"; } else
{ finalMedianSelection = "NO"; }

            selectImage("PreviewImage");
                        saveAs("Tiff", dir2Subfolder + name1WithoutExtension +
outputROIname1 + "_" + channelName + "_TH.tif");
            close();
            break;
        } else {
```

```
            finalAlgo = currentAlgo;
            finalRadius = currentRadius;
                        if (isMedian) { finalMedianSelection = "YES"; } else
{ finalMedianSelection = "NO"; }
        }
    }


    result = newArray(finalAlgo, finalMedianSelection, finalRadius);
    return result;
}
```

3.2.4. Applicazione Maschere e Ricomposizione Preliminare

Una volta finalizzate le tre maschere (una per ogni canale):

1. **Normalizzazione:** Le maschere binarie _TH.tif vengono normalizzate (i loro valori vengono divisi per 255) per ottenere un range da 0 a 1.
2. **Applicazione maschera:** Tramite il comando *imageCalculator* (operazione "Multiply"), l'immagine del canale invertito _BN.tif viene moltiplicata per la sua maschera normalizzata. Questo processo "cancella" (imposta a zero) tutti i pixel che non sono stati identificati come segnale positivo.
3. **Creazione composito:** I tre canali "puliti" _DEF.tif vengono uniti in una nuova immagine a colori utilizzando il comando *Merge Channels...*. L'utente può scegliere la mappatura dei colori; l'opzione di default (G) assegna:
    - **Giallo:** Canale 1 (es. CD3)
    - **Magenta:** Canale 3 (es. CD20)
    - **Blu:** Canale 2 (NUCLEI)

Codice sviluppato:

```
// Process CH1 (CD3)
```

```
resCH1 = processChannel(outputCH1name1, "CH1 (" + outputCH1name1 + ")", "Yen",
15);
algoCH1 = resCH1[0];
medianAppliedCH1 = resCH1[1];
medianRadiusCH1 = resCH1[2];

// Process CH2 (Nuclei)
resCH2 = processChannel(outputCH2name1, "CH2 (" + outputCH2name1 + ")",
"Moments", 0);
algoCH2 = resCH2[0];
medianAppliedCH2 = resCH2[1];
medianRadiusCH2 = resCH2[2];

// Process CH3 (CD20)
resCH3 = processChannel(outputCH3name1, "CH3 (" + outputCH3name1 + ")", "Yen",
18);
algoCH3 = resCH3[0];
medianAppliedCH3 = resCH3[1];
medianRadiusCH3 = resCH3[2];

print("All the binary masks are processed.");

// NORMALIZE AND SAVE THRESHOLDED MASKS
run("Close All");

open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_TH.tif");
run("Divide...", "value=255.000");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_TH.tif");
run("Close All");

open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_TH.tif");
run("Divide...", "value=255.000");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_TH.tif");
run("Close All");

open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_TH.tif");
run("Divide...", "value=255.000");
```

```
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_TH.tif");
run("Close All");

// Overlap the binary masks (Create _DEF.tif files)

// CH1
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_BN.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_TH.tif");
imageCalculator("Multiply create", name1WithoutExtension + outputROIname1 + "_"
+ outputCH1name1 + "_BN.tif", name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_TH.tif");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_DEF.tif");
run("Close All");

// CH2
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_BN.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_TH.tif");
imageCalculator("Multiply create", name1WithoutExtension + outputROIname1 + "_"
+ outputCH2name1 + "_BN.tif", name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_TH.tif");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_DEF.tif");
run("Close All");

// CH3
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_BN.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_TH.tif");
imageCalculator("Multiply create", name1WithoutExtension + outputROIname1 + "_"
+ outputCH3name1 + "_BN.tif", name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_TH.tif");
saveAs("Tiff", dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_DEF.tif");
run("Close All");

// Merge channels
```

```
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_DEF.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_DEF.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_DEF.tif");

Dialog.create("Choose color scheme for merged channels");
Dialog.addMessage("CH1 = " + outputCH1name1 +
                  "\nCH2 = " + outputCH2name1 +
                  "\nCH3 = " + outputCH3name1);
Dialog.addMessage("Select a color scheme:\n" +
                  "A: CH1=green, CH2=blue, CH3=red\n" +
                  "B: CH1=gray, CH2=blue, CH3=yellow\n" +
                  "C: CH1=magenta, CH2=blue, CH3=cyan\n" +
                  "D: CH1=cyan, CH2=blue, CH3=magenta\n" +
                  "E: CH1=red, CH2=blue, CH3=green\n" +
                  "F: CH1=magenta, CH2=blue, CH3=yellow\n" +
                  "G: CH1=yellow, CH2=blue, CH3=magenta");
Dialog.addChoice("Scheme:", newArray("A","B","C","D","E","F","G"), "G");
Dialog.show();
selectionString2 = Dialog.getChoice();

// MERGE CHANNELS

if(selectionString2 == "A"){
     run("Merge Channels...", "c1=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH3name1 + "_DEF.tif] c2=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH1name1 + "_DEF.tif] c3=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH2name1 + "_DEF.tif] create");
} else if (selectionString2 == "B"){
     run("Merge Channels...", "c3=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH2name1 + "_DEF.tif] c4=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH1name1 + "_DEF.tif] c7=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH3name1 + "_DEF.tif] create");
} else if (selectionString2 == "C"){
     run("Merge Channels...", "c3=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH2name1 + "_DEF.tif] c5=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH3name1 + "_DEF.tif] c6=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH1name1 + "_DEF.tif] create");
} else if (selectionString2 == "D"){
     run("Merge Channels...", "c3=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH2name1 + "_DEF.tif] c5=[" + name1WithoutExtension + outputROIname1
```

```
+ "_" + outputCH1name1 + "_DEF.tif] c6=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH3name1 + "_DEF.tif] create");
} else if (selectionString2 == "E"){
     run("Merge Channels...", "c1=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH1name1 + "_DEF.tif] c2=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH3name1 + "_DEF.tif] c3=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH2name1 + "_DEF.tif] create");
} else if (selectionString2 == "F"){
     run("Merge Channels...", "c3=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH2name1 + "_DEF.tif] c6=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH1name1 + "_DEF.tif] c7=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH3name1 + "_DEF.tif] create");
} else if (selectionString2 == "G"){
     run("Merge Channels...", "c3=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH2name1 + "_DEF.tif] c6=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH3name1 + "_DEF.tif] c7=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH1name1 + "_DEF.tif] create");
} else {
     run("Merge Channels...", "c1=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH3name1 + "_DEF.tif] c2=[" + name1WithoutExtension + outputROIname1
+ "_" + outputCH1name1 + "_DEF.tif] c3=[" + name1WithoutExtension +
outputROIname1 + "_" + outputCH2name1 + "_DEF.tif] create");
}
run("Stack to RGB");
rename("Composite_to_Clean");
```

3.2.5. Pulizia Manuale Finale degli Artefatti

Questa è la seconda fase di intervento manuale, progettata per rimuovere artefatti macroscopici (es. pieghe del tessuto, segnale aspecifico) che la segmentazione automatica non ha potuto escludere:

1. **Selezione artefatti:** La macro (funzione “applyChannelCleaning”) chiede all'utente se desidera pulire i canali CH1 e CH3.
2. **Disegno ROI:** L'utente disegna direttamente sull'immagine composita a colori usando lo strumento "freehand" per delineare tutte le aree da rimuovere (tenendo premuto SHIFT per selezioni multiple).

3. **Creazione e applicazione maschere di pulizia:** Per ogni canale da pulire, lo script genera una maschera binaria (invertita e normalizzata) basata su queste selezioni manuali. Successivamente tramite il comando *imageCalculator,* tale maschera viene applicata matematicamente al file immagine corrispondente (DEF.tiff), determinando la soppressione del segnale nelle aree identificate.

Codice sviluppato:

```
// Function for manual cleaning
function applyManualCleaning(channelColor, channelName) {
    selectImage("Composite_to_Clean");
    selectionManual = getString("Enter YES to manually remove areas from the " +
channelColor + " channel (" + channelName + ")", "NO");

    if (selectionManual == "YES") {
        print("Draw all regions you want to remove from the " + channelColor + "
channel.");
        setTool("freehand");
           waitForUser("Draw areas to remove on the " + channelColor + " (" +
channelName + ") channel.\nHold SHIFT to add multiple regions.\nClick OK when
finished.");

        if (selectionType() != -1) {
            // 1. Create Mask from user drawing
            run("Create Mask");
            run("Invert");
              run("Divide...", "value=255.000"); // Mask is now 0 (remove) and 1
(keep)
            rename("CleaningMask");

            // 2. Open the specific channel _DEF file again
                 defImageName = name1WithoutExtension + outputROIname1 + "_" +
channelName + "_DEF.tif";
             // Check if image is already open (from "keep" in merge), otherwise
open it
            if (!isOpen(defImageName)) {
                open(dir2Subfolder + defImageName);
            }
            selectImage(defImageName);
```

```
            // 3. Apply Mask to the specific channel
            imageCalculator("Multiply create", defImageName, "CleaningMask");

            // 4. Overwrite the _DEF file with the cleaned version
            saveAs("Tiff", dir2Subfolder + defImageName);

            // 5. Cleanup temp images
            close("CleaningMask");
            // We can close the result image, we will re-open for final merge
            close();
            print("Cleaning applied to " + channelName);

            return "YES";
        } else {
            print("No selection made for " + channelName);
            return "NO";
        }
    }
    return "NO";
}
```

3.2.6. Applicazione Finale, Salvataggio e Logging

Il flusso di lavoro terminale si articola nei seguenti passaggi:

1. **Merge finale:** I canali CH1, CH2, CH3, ora definitivamente puliti, vengono ri-fusi nell'immagine composita finale.
2. **Salvataggio immagine:** L'immagine finale viene salvata in formato PNG.
3. **Scrittura del log:** La macro scrive nel file di log un riepilogo di tutti i parametri scelti: i nomi dei canali, gli algoritmi di thresholding utilizzati, i raggi dei filtri mediani e la conferma dell'avvenuta pulizia manuale. Questo garantisce la tracciabilità e la riproducibilità dell'intero processo.
4. **Pulizia automatica:** Per ottimizzare lo spazio di archiviazione e mantenere ordinata la directory di output, lo script esegue un ciclo di pulizia finale (Garbage Collection). L'algoritmo scansiona tutti i file presenti nella cartella di destinazione e applica un filtro logico di tipo "whitelist". Vengono preservati esclusivamente:

- I file immagine dei singoli canali definitivi.
- L'immagine composita finale.
- Il file di log.

Un approccio metodologico speculare è stato adottato per la gestione degli altri pannelli immunoistochimici.

Codice sviluppato:

```
// PERFORM MANUAL CLEANING
// Clean Yellow (CH1 - CD3)
manualAppliedYellow = applyManualCleaning("YELLOW", outputCH1name1);

// Clean Magenta (CH3 - CD20)
manualAppliedMagenta = applyManualCleaning("MAGENTA", outputCH3name1);

run("Close All");

// FINAL MERGE
// Re-open the (potentially cleaned) _DEF files
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH1name1 + "_DEF.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_DEF.tif");
open(dir2Subfolder + name1WithoutExtension + outputROIname1 + "_" +
outputCH3name1 + "_DEF.tif");

// Re-Merge using Scheme G (Yellow, Blue, Magenta)
run("Merge Channels...", "c3=[" + name1WithoutExtension + outputROIname1 + "_" +
outputCH2name1 + "_DEF.tif] c6=[" + name1WithoutExtension + outputROIname1 + "_"
+ outputCH3name1 + "_DEF.tif] c7=[" + name1WithoutExtension + outputROIname1 +
"_" + outputCH1name1 + "_DEF.tif] create");
run("Stack to RGB");

// FINAL DIALOG AND LOG
Dialog.create("Processing!");
Dialog.addMessage("Please, click here OK but then wait 2 minutes for automatic
saving!");
Dialog.show();
```

```
saveAs("PNG", dir2Subfolder + name1WithoutExtension + outputROIname1 +
"_Composite.png");

logText = "";
logText += "Image: " + name1 + "\n\n";
logText += "ROI Name: " + outputROIname1 + "\n\n";
logText += "Output Directory: " + dir2Subfolder + "\n\n";
logText += "CH1 (" + outputCH1name1 + " - Yellow):\n Threshold: " + algoCH1 +
"\n Median: " + medianAppliedCH1 + "\n Radius: " + medianRadiusCH1 + "\n Manual
Cleaning: " + manualAppliedYellow + "\n\n";
logText += "CH2 (" + outputCH2name1 + " - Blue):\n Threshold: " + algoCH2 + "\n
Median: " + medianAppliedCH2 + "\n Radius: " + medianRadiusCH2 + "\n\n";
logText += "CH3 (" + outputCH3name1 + " - Magenta):\n Threshold: " + algoCH3 +
"\n Median: " + medianAppliedCH3 + "\n Radius: " + medianRadiusCH3 + "\n Manual
Cleaning: " + manualAppliedMagenta + "\n\n";

File.saveString(logText, logFile);
print("Processing log saved.");

// CLEANUP INTERMEDIATE FILES
// Deletes everything except _DEF.tif, _Composite.png, and log files.
print("Cleaning up intermediate files...");
fileList = getFileList(dir2Subfolder);
for (i = 0; i < fileList.length; i++) {
    filename = fileList[i];

    // Check if the file is one of the ones we want to KEEP
    isDef = endsWith(filename, "_DEF.tif");
    isComposite = endsWith(filename, "_Composite.png");
    isLog = endsWith(filename, "_processing_log.txt");

    // If it's NOT one of the keepers, delete it
    if (!isDef && !isComposite && !isLog) {
        ok = File.delete(dir2Subfolder + filename);
    }
}

print("Intermediate files deleted.");
print("Now you can close!");
run("Close All");
```

4. Script Groovy per segmentazione singole cellule

Il processo di segmentazione consiste nell'identificazione e isolamento di particolari regioni di interesse all'interno di immagini istologiche.

Nel contesto di questa tesi poniamo particolare interesse nell'individuazione e segmentazione di cellule in *QuPath*.

Segmentazione automatica

Attraverso l'utilizzo di algoritmi avanzati di segmentazione, *QuPath* permette l'automatizzazione del riconoscimento di diverse componenti cellulari e strutture anatomiche, semplificando i procedimenti di analisi e migliorandone l'efficienza e la precisione.

4.1. Cellpose

Per affrontare le complessità legate alla segmentazione cellulare in immagini microscopiche eterogenee, è fondamentale fare riferimento allo stato dell'arte rappresentato da *Cellpose*. Un algoritmo di segmentazione "generalista" basato sul deep learning, progettato per operare su una vasta gamma di tipi di immagini senza la necessità di riaddestramento del modello o di aggiustamento dei parametri. A differenza dei metodi classici (come l'algoritmo *Watershed*), che spesso falliscono quando le cellule sono densamente impacchettate o presentano profili di intensità non uniformi, *Cellpose* introduce un approccio innovativo basato sulla previsione di campi di flusso vettoriale (vector flow fields). Il funzionamento del modello si basa su tre principi chiave:

1. **Rappresentazione tramite flussi:** La rete neurale non si limita a classificare i pixel, ma predice gradienti spaziali orizzontali e verticali. Attraverso un processo di tracciamento del gradiente, i

pixel vengono "guidati" verso il centro della cellula a cui appartengono, permettendo di separare con precisione anche cellule adiacenti o sovrapposte.

2. **Architettura avanzata:** Il modello utilizza un'architettura U-Net modificata che integra blocchi residui (residual blocks) e un vettore di stile globale (style vector), il quale permette alla rete di adattare l'elaborazione alle caratteristiche specifiche di ogni immagine.
3. **Addestramento generalista:** La robustezza dell'algoritmo deriva dal suo addestramento su un dataset estremamente variegato, contenente oltre 70.000 oggetti segmentati provenienti da diverse modalità di microscopia e persino da immagini non biologiche, garantendo elevate capacità di generalizzazione [8].

4.1.1. Segmentazione tramite algoritmo Cellpose

Per l'identificazione e la fenotipizzazione delle popolazioni linfocitarie B (CD20) e T (CD3), è stato sviluppato un protocollo di analisi automatizzato all'interno della piattaforma QuPath. La procedura utilizza l'algoritmo di deep learning *Cellpose* integrato in un flusso di lavoro sequenziale, progettato per massimizzare la specificità del rilevamento adattando i parametri morfologici a ciascun tipo cellulare. Il flusso di lavoro si articola nelle seguenti fasi:

1. **Inizializzazione e definizione dell'area di analisi:** Il protocollo acquisisce preliminarmente i dati di calibrazione dell'immagine per garantire misurazioni metriche precise. Successivamente, verifica la presenza di una selezione attiva da parte dell'utente; in assenza di una Regione di Interesse (ROI) specifica, il sistema genera automaticamente un'annotazione che comprende l'intera immagine, assicurando che nessuna porzione di tessuto venga esclusa dall'analisi. Eventuali rilevamenti precedenti vengono rimossi per garantire un ambiente di lavoro pulito.
2. **Segmentazione sequenziale adattiva:** A differenza degli approcci standard che utilizzano parametri generici per tutte le cellule, questo metodo applica una strategia a due passaggi distinti, ottimizzati per le diverse caratteristiche morfologiche delle popolazioni target:
   - **Rilevamento linfociti B (canale CD20):** In una prima fase, l'algoritmo analizza esclusivamente il canale relativo al marcatore CD20. Il modello di segmentazione viene configurato con un parametro di diametro cellulare atteso più elevato (30 pixel), calibrato per intercettare correttamente le dimensioni tipicamente maggiori o la tendenza all'aggregazione

dei linfociti B follicolari. Alle cellule identificate viene assegnata immediatamente la classificazione "CD20".

- **Rilevamento linfociti T (canale CD3):** I risultati della prima fase vengono memorizzati temporaneamente e il sistema procede con una seconda scansione indipendente sul canale CD3. In questo caso, il parametro del diametro viene ridotto (15 pixel) per rispecchiare la morfologia più compatta e le dimensioni ridotte tipiche dei linfociti T. Alle cellule rilevate in questo passaggio viene assegnata la classificazione “CD3".

3. **Integrazione dei dati e fenotipizzazione:** Al termine delle due scansioni indipendenti, le popolazioni cellulari rilevate vengono unite in un'unica mappa di segmentazione globale. Questo approccio sequenziale garantisce che ogni cellula venga identificata con i parametri ideali per la sua tipologia, riducendo drasticamente gli errori di sovra-segmentazione (cellule grandi spezzate in due) o sotto-segmentazione (cellule piccole ignorate).
4. **Estrazione delle caratteristiche quantitative:** Come fase conclusiva, per ogni singolo elemento cellulare identificato, il sistema calcola e associa un set completo di descrittori quantitativi. Questi includono parametri morfometrici (come area, perimetro e circolarità) e valori di intensità media del segnale per tutti i canali di fluorescenza disponibili, fornendo il dataset numerico necessario per le successive analisi statistiche.

Nonostante la robustezza teorica del modello *Cellpose*, basato sulla predizione di campi di flusso vettoriale per la segmentazione generalista, l'applicazione sperimentale del protocollo sopra descritto sui campioni in esame ha prodotto risultati non conformi agli standard di accuratezza richiesti dallo studio. Nello specifico, il modello di Deep Learning ha mostrato limitazioni nella gestione della variabilità intrinseca del tessuto patologico, fallendo in alcuni casi nel discriminare correttamente i confini cellulari in aree ad alta densità o presentando artefatti di segmentazione non correggibili tramite i soli parametri di diametro. La natura "black-box" della rete neurale ha impedito un fine-tuning granulare sui singoli canali di fluorescenza, rendendo complesso l'adattamento del modello alle specificità dei marcatori CD3 e CD20. A fronte di tali criticità, e al fine di garantire una riproducibilità rigorosa del dato quantitativo, si è optato per l'abbandono della strategia basata sul Deep Learning in favore di un approccio deterministico alternativo. La scelta è ricaduta sull'implementazione di un protocollo personalizzato basato sull'algoritmo classico *Watershed.*

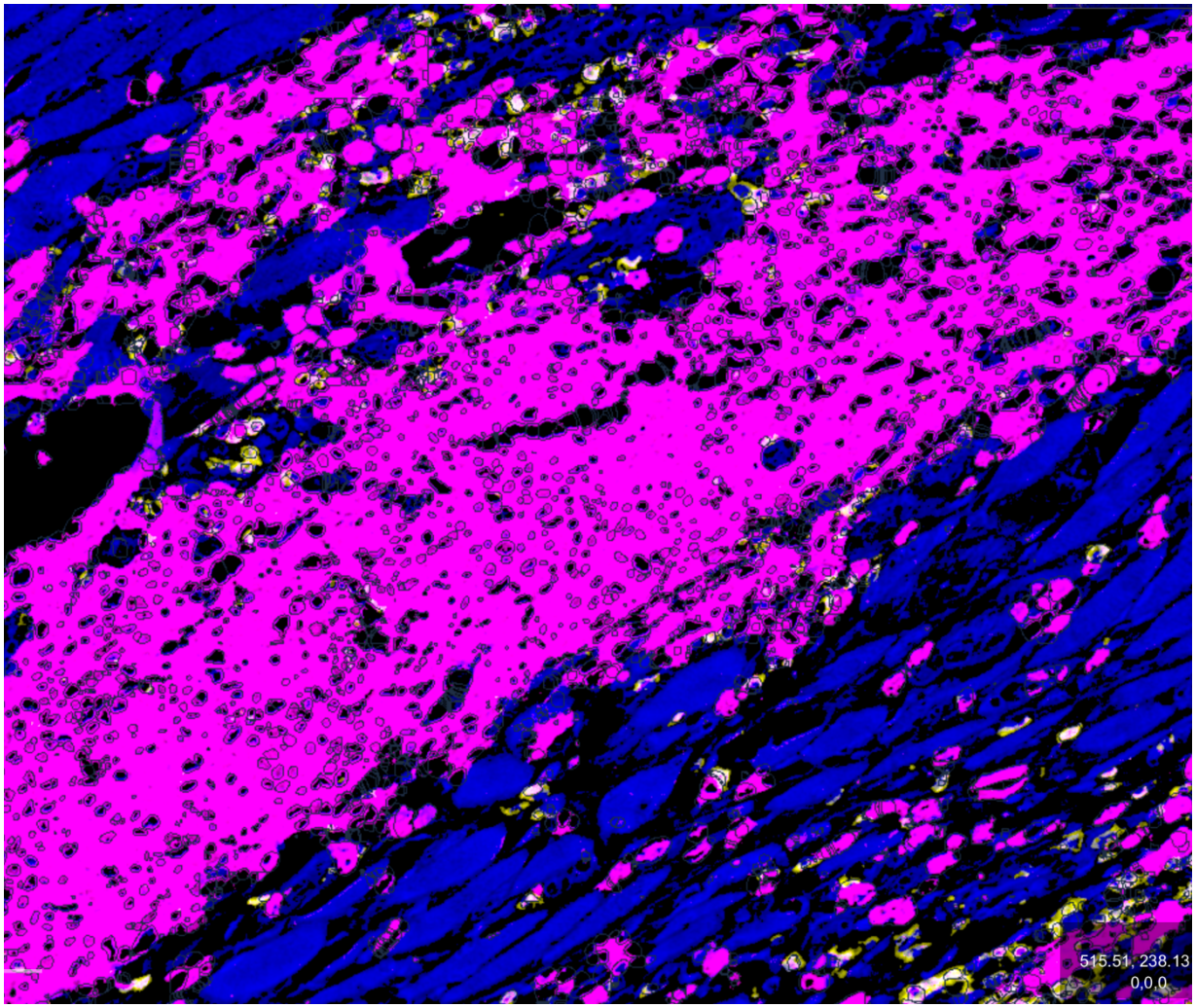

Figura 4.1 - Segmentazione algoritmo Cellpose

4.2. Watershed

L'algoritmo *Watershed* fonda il suo funzionamento su una metafora geografica, interpretando l'immagine digitale in scala di grigi come un rilievo topografico tridimensionale in cui i valori di intensità dei pixel corrispondono all'elevazione: le aree scure rappresentano le valli, mentre quelle chiare costituiscono i picchi. Il principio operativo del modello classico si articola attraverso un processo di "inondazione" che ha inizio dai minimi locali del rilievo, i quali fungono da bacini di

raccolta iniziali. Partendo dal minimo di altezza inferiore, un'acqua "virtuale" inizia a salire gradualmente, riempiendo i bacini di utenza circostanti. Durante questa fase di immersione, nel momento in cui si verifica il rischio che le acque provenienti da bacini differenti convergano, l'algoritmo erige delle barriere digitali nei punti di incontro, definite "linee spartiacque" (Watershed Lines o WL) o dighe. Il processo termina quando l'acqua raggiunge il picco massimo del rilievo, restituendo un risultato finale in cui ogni bacino di utenza, corrispondente a uno specifico segmento o oggetto, risulta completamente circondato e delimitato da tali linee di separazione[9].

4.2.1. Segmentazione tramite algoritmo Watershed

Lo script esegue un flusso di lavoro automatizzato per l'identificazione, la classificazione e l'analisi spaziale di due popolazioni cellulari distinte (CD3 e CD20) all'interno di regioni rettangolari di area uniforme denominate “Tile”, in cui è stata suddivisa l’intera ROI. Il processo si articola nelle seguenti fasi operative:

1. **Inizializzazione e preparazione dell'area di lavoro:** Come passaggio preliminare, l'algoritmo verifica l'esistenza delle annotazioni di lavoro (i "Tiles"). In caso positivo, procede a una pulizia preventiva dell'immagine, rimuovendo qualsiasi oggetto rilevato (detection) precedentemente classificato come "Magenta" o "Yellow". Questo garantisce che l'analisi venga eseguita su un set di dati pulito, evitando duplicazioni o sovrapposizioni con esecuzioni precedenti.
2. **Strategia di segmentazione iterativa per canale:** Il cuore dell'elaborazione avviene attraverso un ciclo iterativo che analizza ogni singolo "Tile" in modo indipendente. Per ogni regione, viene applicata una strategia di segmentazione in due step basata sull'algoritmo *Watershed*:
   - **Rilevamento popolazione CD3 (canale Yellow):** Viene eseguita la segmentazione sul canale specifico per il marcatore CD3. L'algoritmo utilizza parametri ottimizzati per questa popolazione (raggio mediano e soglie di intensità specifici). Le cellule identificate vengono temporaneamente salvate in memoria e classificate come "Yellow".
   - **Rilevamento popolazione CD20 (canale Magenta):** Successivamente, sullo stesso Tile, viene eseguita un secondo step di segmentazione sul canale CD20. In questo caso, i parametri di segmentazione (come l'area minima e il raggio mediano) sono differenti, riflettendo le diverse

caratteristiche morfologiche delle cellule B rispetto alle T. Le cellule risultanti vengono classificate come "Magenta".

- **Fusione dei risultati:** Al termine delle due scansioni, lo script combina i risultati, assicurando che entrambe le popolazioni (Yellow e Magenta) coesistano all'interno della stessa regione di interesse (Tile) senza sovrascriversi a vicenda.

3. **Consolidamento della gerarchia:** Una volta completata l'analisi su tutti i Tiles, lo script esegue un'operazione di gestione della gerarchia degli oggetti. Tutte le cellule rilevate vengono "promosse" o confermate come oggetti figli dell'immagine o delle annotazioni principali. Questo passaggio è tecnico ma fondamentale per permettere alle funzioni successive di calcolare le distanze tra cellule appartenenti a Tiles diversi o adiacenti.
4. **Analisi morfometrica e controllo qualità (filtro dimensionale):** A tutte le cellule identificate vengono aggiunte misurazioni dettagliate sulla forma (area, circolarità, solidità, diametri massimi e minimi). Successivamente, viene applicato un filtro di controllo qualità rigido:
   - L'algoritmo esamina l'area del nucleo di ogni singola cellula rilevata.
   - Viene imposta una soglia minima (cut-off) di 10.0 μm^2.
   - Qualsiasi rilevamento che presenti un nucleo con superficie inferiore a tale soglia, o privo di dati misurabili, viene considerato un artefatto o un frammento cellulare e viene automaticamente rimosso dall'analisi.
5. **Analisi spaziale:** Nella fase conclusiva, lo script calcola le distanze tra i centroidi di tutte le rilevazioni rimaste. Infine, viene risolta la gerarchia degli oggetti per consolidare le relazioni spaziali genitore-figlio, rendendo i dati pronti per l'esportazione e l'analisi statistica finale.

Il confronto tra le metodiche ha premiato l'utilizzo dell'algoritmo *Watershed* rispetto alla rete neurale *Cellpose*, grazie a una maggiore fedeltà nel riconoscimento dei marcatori di membrana specifici (CD3/CD20). Tuttavia, l'approccio classico basato sullo spartiacque ha rivelato i suoi limiti topologici in presenza di aggregati di grandi dimensioni. Poiché il *Watershed* si basa sull'interpretazione dell'intensità del segnale come rilievo topografico, le aree interne ai grossi cluster cellulari caratterizzate da un segnale di fluorescenza piatto e uniforme (assenza di "valli" profonde) vengono erroneamente interpretate come un unico bacino di utenza, impedendo la corretta separazione dei singoli nuclei al loro interno.

Codice sviluppato:

```
/**
 * Automated script for differential segmentation and spatial analysis of
  *  CD3+  (Yellow)  and  CD20+  (Magenta)  lymphocyte  populations  within  'Tile'
annotations.
 * * Workflow:
 * 1. Identify ROI tiles.
 * 2. Clear previous specific detections.
 * 3. Perform two-pass Watershed segmentation (CD3 then CD20) with population-
specific parameters.
 * 4. Merge results and promote objects to the root hierarchy.
 * 5. Apply Quality Control (QC) filtering based on nuclear area.
 * 6. Compute spatial metrics (distances).
 */

// 1. INITIALIZATION AND ROI IDENTIFICATION
// Find all annotation objects named "Tile" without an existing class
def  tiles  =  getAnnotationObjects().findAll  {  it.getPathClass()  ==  null  &&
it.getName().startsWith("Tile") }

if (tiles.isEmpty()) {
    print "Error: No 'Tile' annotations found."
    return
}
print "Ready to process detections in ${tiles.size()} tiles..."

// 2. CLEANUP PREVIOUS DATA
// Identify and remove existing 'Magenta' (CD20) or 'Yellow' (CD3) detections
// to ensure a clean analysis run.
def allOriginalBlobs = getDetectionObjects().findAll {
           it.getPathClass() == getPathClass("Magenta") || it.getPathClass() ==
getPathClass("Yellow")
}

// Remove identified objects
removeObjects(allOriginalBlobs, true)

// 3. BATCH PROCESSING
// Iterate through each identified tile
tiles.each { tile ->
    // Select the current tile as the parent object for detection
    selectObjects(tile)
```

```
    // PHASE A: CD3 DETECTION (YELLOW
    // Run Watershed Cell Detection on the CD3 channel
    // Parameters optimized for T-cells (smaller radius)
    runPlugin('qupath.imagej.detect.cells.WatershedCellDetection', """
        {
          "detectionImage": "CD3",
          "requestedPixelSizeMicrons": 0.5,
          "backgroundRadiusMicrons": 8.0,
          "medianRadiusMicrons": 4.0,
          "cellExpansionMicrons": 0.01,
          "sigmaMicrons": 1.5,
          "minAreaMicrons": 20.0,
          "maxAreaMicrons": 200.0,
          "threshold": 0.1,
          "includeNuclei": true,
          "smoothBoundaries": true,
          "makeMeasurements": false
        }
    """)

    // Retrieve the newly created detections (children of the tile)
        def newYellowDetections = new ArrayList(tile.getChildObjects().findAll
{ it.isDetection() })
    // Assign the 'Yellow' class to CD3+ cells
    newYellowDetections.each { it.setPathClass(getPathClass("Yellow")) }


    // PHASE B: CD20 DETECTION (MAGENTA
    // Run Watershed Cell Detection on the CD20 channel
     // Note: Running this plugin overwrites the current children of the tile in
the viewer,
    // which is why we stored the Yellow detections in a list above.
    // Parameters optimized for B-cells (larger median radius and min area)
    runPlugin('qupath.imagej.detect.cells.WatershedCellDetection', """
        {
          "detectionImage": "CD20",
          "requestedPixelSizeMicrons": 0.5,
          "backgroundRadiusMicrons": 8.0,
          "medianRadiusMicrons": 5.0,
          "cellExpansionMicrons": 0.01,
          "sigmaMicrons": 1.5,
          "minAreaMicrons": 40.0,
          "maxAreaMicrons": 200.0,
```

```
            "threshold": 0.1,
            "includeNuclei": true,
            "smoothBoundaries": true,
            "makeMeasurements": false
        }
    """)

     // Retrieve the new detections (currently the only children visible on the
tile)
       def newMagentaDetections = new ArrayList(tile.getChildObjects().findAll
{ it.isDetection() })
    // Assign the 'Magenta' class to CD20+ cells
    newMagentaDetections.each { it.setPathClass(getPathClass("Magenta")) }


    // PHASE C: MERGE RESULTS
    // The tile currently contains only Magenta detections.
    // Add the stored Yellow detections back to the tile to combine populations.
    addObjects(newYellowDetections)
}

// 4. HIERARCHY REORGANIZATION
// Promote detections to the Root object to facilitate global distance
calculations
// across tile boundaries.

// Collect all final detections (both Magenta and Yellow)
def allFinalDetections = new ArrayList(getDetectionObjects())

// (Optional: List tiles if needed for removal, though not removed in this
logic)
def allTiles = getAnnotationObjects().findAll { it.getName() != null &&
it.getName().startsWith("Tile") }

// Add all detections to the image hierarchy root
addObjects(allFinalDetections)


// 5. MORPHOMETRY AND QUALITY CONTROL (QC)
selectDetections()
// Compute morphological features
addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER", "NUCLEUS_CELL_RATIO")
```

```
// QC FILTER: NUCLEUS AREA
// Define strict threshold for minimum nucleus size
def minNucleusArea = 10.0

// Identify artifacts or fragments with nuclei smaller than the threshold
def detectionsToRemove = getDetectionObjects().findAll {
    // Retrieve the specific measurement
    def nucleusArea = it.getMeasurementList().get("Nucleus: Area µm^2")
    // Flag for removal if measurement is missing or below threshold
    return nucleusArea == null || nucleusArea < minNucleusArea
}

// Execute filtering
if (!detectionsToRemove.isEmpty()) {
    print "QC FILTER: Detected ${detectionsToRemove.size()} objects with nucleus
< ${minNucleusArea} µm²."
    removeObjects(detectionsToRemove, true)
    print "QC FILTER: Small artifacts removed."
} else {
    print "QC FILTER: All detections passed the area threshold."
}


// 6. SPATIAL ANALYSIS
// Select valid detections
selectDetections()
// Compute Euclidean distances between centroids of all detections
detectionCentroidDistances()
// Finalize and resolve object hierarchy
resolveHierarchy()

print "Script completed: Detections filtered and distances calculated."
```

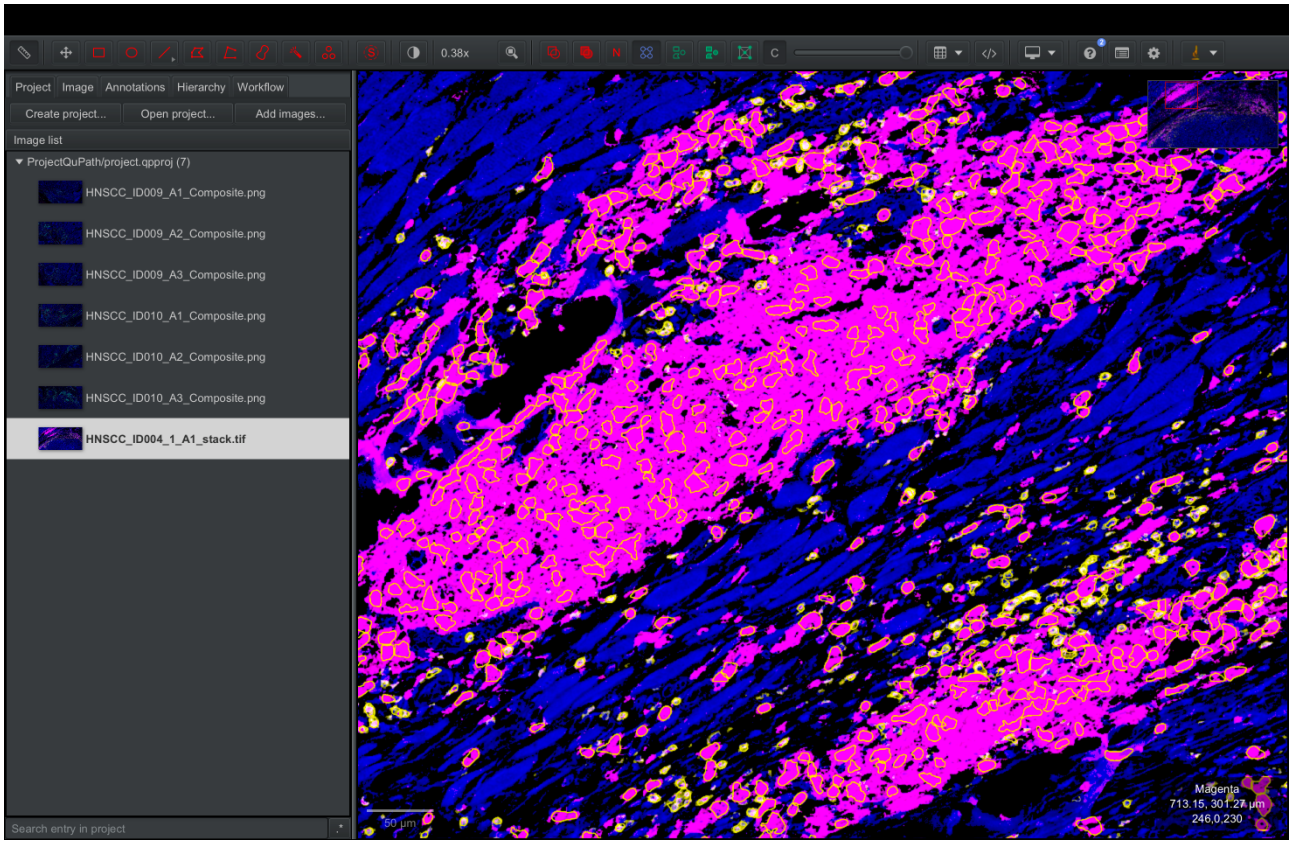

Figura 4.2 - Segmentazione algoritmo Watershed

4.3. Segmentazione manuale

In ultima analisi, per superare le criticità residue riscontrate sia con l'approccio di Deep Learning (*Cellpose*) che con quello morfologico (*Watershed*), è stato implementato un protocollo di rifinitura manuale assistita. Tale procedura è stata introdotta per correggere puntualmente gli errori di segmentazione, in particolare la mancata separazione degli aggregati cellulari complessi, garantendo così la massima accuratezza del dato finale.

4.3.1. Tiling e Classificazione dei Pixel

Per gestire l'analisi di immagini istologiche di grandi dimensioni e identificare le aree positive ai marcatori CD20 e CD3, è stato sviluppato un flusso di lavoro automatizzato in *QuPath* che combina la suddivisione dell'immagine (*Tiling*) con l'uso di Classificatori di Pixel (*Pixel Classifiers*). La procedura si articola in quattro fasi operative principali:

1. **Calibrazione e suddivisione spaziale (tiling):** In via preliminare, lo script imposta forzatamente la dimensione del pixel (0.2526 μm) per garantire la coerenza delle misurazioni metriche. Successivamente, le macro-regioni di interesse identificate come "Region" vengono suddivise in unità di analisi più piccole e gestibili, denominate "Tiles" (tasselli), ciascuna delle dimensioni di 510x510 μm, generando in media 20 Tiles per ciascuna ROI. Questa operazione di *tiling* è fondamentale per ottimizzare il carico computazionale e permettere un'elaborazione locale più precisa.
2. **Segmentazione fenotipica iterativa:** Per ogni singolo Tile generato, l'algoritmo esegue due passaggi sequenziali di segmentazione basati su classificatori di pixel pre-addestrati:
   - **Segmentazione CD20 (popolazione B):** Viene applicato il classificatore per il canale CD20. L'algoritmo identifica i pixel positivi e li converte in oggetti vettoriali (annotazioni) di classe "Magenta". Viene applicato un filtro dimensionale che scarta oggetti con area inferiore a 10.0 micrometri per ridurre il segnale di fondo ed escludere oggetti di dimensioni non compatibili con una cellula.
   - **Segmentazione CD3 (popolazione T):** Analogamente, viene applicato il classificatore al canale CD3 per generare oggetti di classe "Yellow" corrispondenti ai linfociti T, mantenendo le stesse soglie di filtraggio. Contestualmente alla creazione, vengono calcolati i descrittori morfologici preliminari per ogni oggetto.
3. **Conversione tipologica degli oggetti:** Una particolarità tecnica di questo protocollo risiede nella gestione delle classi di oggetti *QuPath*. Inizialmente, le aree segmentate vengono generate come "annotazioni" (regioni). Tuttavia, per abilitare le successive analisi di conta e densità, lo script esegue una conversione programmatica: identifica tutte le annotazioni "Magenta" e "Yellow" appena create e le trasforma in "Detections" (rilevamenti). Questo passaggio è cruciale per trattare le aree positive come entità biologiche distinte e conteggiabili.
4. **Morfometria finale e pulizia:** Nella fase conclusiva, a tutti i nuovi oggetti "Detection" vengono associati parametri morfometrici dettagliati, inclusi area, perimetro, circolarità, solidità e diametri

(massimo e minimo). Infine, le annotazioni temporanee originali vengono rimosse dall'immagine per evitare duplicazioni, lasciando visibili solo i rilevamenti finali pronti per l'analisi statistica.

Codice sviluppato:

```
import qupath.lib.objects.PathObjects

// Set pixel size (if not already set from metadata)
setPixelSizeMicrons(0.252600, 0.252600)

// Create tiles
selectObjectsByClassification("Region*");
runPlugin('qupath.lib.algorithms.TilerPlugin',
'{"tileSizeMicrons":510.0,"trimToROI":true,"makeAnnotations":true,"removeParentA
nnotation":false}')

// Find the newly created tiles (unclassified annotations)
tiles = getAnnotationObjects().findAll{it.getPathClass() == null}
print tiles

// Create Magenta ANNOTATIONS
tiles.each { tile ->
    selectObjects(tile)
         // Note: addPixelClassifierMeasurements is technically optional if
createAnnotations is run immediately,
    // but useful if you need measurements on the tile itself.
    addPixelClassifierMeasurements("CD20_full_thr1", "CD20_full_thr1")
     createAnnotationsFromPixelClassifier("CD20_full_thr1", 10.0, 0.0, "SPLIT",
"IGNORE_EXISTING")
         addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER", "NUCLEUS_CELL_RATIO")
}

// Create Yellow ANNOTATIONS
tiles.each { tile ->
    selectObjects(tile)
    addPixelClassifierMeasurements("CD3_full_thr1", "CD3_full_thr1")
      createAnnotationsFromPixelClassifier("CD3_full_thr1", 10.0, 0.0, "SPLIT",
"IGNORE_EXISTING")
         addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER", "NUCLEUS_CELL_RATIO")
```

```
}

// Convert ALL annotations (Magenta/Yellow) to DETECTIONS

// Find all newly created annotations
def annotsToConvert = getAnnotationObjects().findAll {
        it.getPathClass() == getPathClass("Magenta") || it.getPathClass() ==
getPathClass("Yellow")
}

if (annotsToConvert.isEmpty()) {
    print "Error: No 'Magenta' or 'Yellow' annotations created. Check your Pixel
Classifiers."
    return
}

// Create a list of new detections based on the annotations
def newDetections = annotsToConvert.collect {
    return PathObjects.createDetectionObject(it.getROI(), it.getPathClass())
}

// Calculate measurements for the new detections
selectDetections();
addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER")

// Remove old annotations and add the new detections
removeObjects(annotsToConvert, true)
addObjects(newDetections)

// Final hierarchy resolution and distance calculations
selectDetections();
detectionCentroidDistances()
resolveHierarchy()

print "Step 1 completed: Created ${newDetections.size()} total 'blob'
detections."
```

4.3.2. Algoritmo di tassellatura esagonale adattiva per macro-aggregati

Per risolvere le problematiche di sotto-segmentazione in aree tissutali ad alta densità, è stato implementato un algoritmo personalizzato in linguaggio *Groovy* ("HexGridFromBlobs"). Questo script implementa una logica ibrida e condizionale: preserva le singole cellule correttamente segmentate e interviene esclusivamente sui macro-aggregati (clusters), sostituendoli con una tassellatura esagonale statisticamente calibrata. Il flusso di lavoro computazionale si articola nelle seguenti fasi:

1. **Analisi statistica della distribuzione dimensionale (modal diameter calculation):** Prima di procedere alla segmentazione geometrica, l'algoritmo esegue un'analisi preliminare per determinare la dimensione ottimale della griglia per ciascun canale (es. "Yellow" per CD3, "Magenta" per CD20):
   - **Filtraggio del rumore:** Viene applicata una soglia di area minima (es. 15 μm² per CD3, 30 μm² per CD20) per escludere frammenti e detriti dall'analisi statistica.
   - **Istogramma e smoothing:** Le aree degli oggetti validi vengono distribuite in un istogramma di frequenza (bin size = 5 μm²). Per mitigare il rumore statistico, viene applicata una media mobile (moving average) sui bin adiacenti.
   - **Identificazione della moda:** L'algoritmo identifica il picco dell'istogramma (modal area), che rappresenta la dimensione cellulare più frequente e rappresentativa della popolazione.
   - **Conversione geometrica:** L'area modale viene convertita nel diametro equivalente di un esagono regolare utilizzando la formula inversa dell'area dell'esagono. Questo garantisce che la griglia generata rispecchi la densità di impacchettamento naturale del tessuto.
2. **Logica condizionale di selezione:** L'algoritmo itera all'interno di ciascuna unità di analisi (*Tile*) applicando un criterio dimensionale selettivo per distinguere le cellule singole dagli aggregati:
   - **Cellule singole:** Gli oggetti con area inferiore a 300 μm² vengono considerati correttamente segmentati e mantenuti inalterati nel dataset.
   - **Cluster sovra-confluenti:** Gli oggetti che superano la soglia di 300 μm² vengono identificati come aggregati e sottoposti alla procedura di tassellatura.
3. **Generazione della griglia esagonale (tiling):** Per ogni cluster identificato, l'algoritmo costruisce una griglia virtuale di esagoni che copre l'intera estensione del rettangolo di ingombro (bounding box). La disposizione degli esagoni segue un pattern "a nido d'ape" (staggered grid), ottimizzato

per massimizzare la copertura spaziale isotropa, simulando l'impacchettamento cellulare nei tessuti biologici meglio di una griglia rettangolare.

4. **Intersezione booleana e segmentazione topologica:** Il passaggio critico consiste nell'adattare la griglia astratta alla morfologia reale dell'oggetto biologico:
   - **Intersezione (clipping):** Viene calcolata l'intersezione geometrica tra ogni esagono della griglia e l'area irregolare del blob. Questo processo ritaglia gli esagoni esattamente lungo i confini del rilevamento primario, mantenendo inalterata la forma complessiva dell'aggregato. Al contempo, l'algoritmo riconosce e preserva i "buchi" interni presenti nella struttura originale, escludendoli dalla geometria finale.
   - **Gestione delle discontinuità (splitting):** Se l'intersezione produce poligoni disconnessi (ad esempio, un esagono che attraversa una "baia" vuota nel blob), l'algoritmo rileva le componenti connesse multiple e le separa automaticamente in oggetti distinti.
5. **Sostituzione e misurazione:** Al termine dell'elaborazione, i macro-aggregati originali vengono rimossi dalla gerarchia e sostituiti dai nuovi oggetti esagonali (o frammenti di essi). A ciascun nuovo oggetto vengono associate automaticamente le misurazioni morfometriche standard, rendendoli pronti per le successive analisi di densità e distanza spaziale.

Questo approccio ibrido combina la flessibilità della segmentazione biologica (il contorno esterno del blob) con la regolarità della discretizzazione geometrica, permettendo una stima accurata della densità cellulare anche in assenza di gradienti di membrana risolvibili.

Codice sviluppato:

```
import qupath.lib.objects.PathObjects
import qupath.lib.roi.*
import qupath.lib.regions.ImagePlane
import qupath.lib.geom.Point2
import java.awt.geom.Area
import org.slf4j.LoggerFactory
import qupath.lib.roi.GeometryTools

/**
 * HexGridFromBlobs
 * * Splits blobs into hexagons ONLY if area >= 300 µm².
 * * Maintaining the shape of the original structure.
```

```
 */
class HexGridFromBlobs {

    String channelName
    double diametro
    def logger = LoggerFactory.getLogger("HexGridFromBlobs")

    // Geometric parameters
    double a
    double stepX
    double stepY
    ImagePlane plane
    double pxSize

    // MINIMUM THRESHOLD
    double minBlobAreaToSplit = 300.0

    HexGridFromBlobs(String channelName, double diametroMicron) {
        this.channelName = channelName

        // Retrieve pixel calibration
                                                                def cal =
qupath.lib.gui.QuPathGUI.getInstance().getImageData().getServer().getPixelCalibr
ation()
        this.pxSize = cal.getPixelWidthMicrons()

        if (Double.isNaN(pxSize) || pxSize == 0) {
            this.pxSize = 1.0
            print "Warning: Pixel size not found, assuming 1.0"
        }

        double diametroPixel = diametroMicron / pxSize
        this.diametro = diametroPixel

        this.a = diametro / 2.0
        this.stepX = 1.5 * a
        this.stepY = Math.sqrt(3) * a
        this.plane = ImagePlane.getDefaultPlane()
    }

    void runOnParent(def parentObject) {

        // Find child blobs of the current Tile
```

```
        def blobs = parentObject.getChildObjects().findAll {
            it.isDetection() && it.getPathClass() == getPathClass(channelName)
        }

        if (blobs.isEmpty()) return

        def allHexes = []
        def blobsToRemove = []

        blobs.each { blob ->

            def areaMicrons = blob.getMeasurementList().get("Area µm^2")
            if (areaMicrons == null || Double.isNaN(areaMicrons)) return
            if (areaMicrons < minBlobAreaToSplit) return

            blobsToRemove << blob

            def roi = blob.getROI()
            def blobArea = new Area(roi.getShape())

            double xMin = roi.getBoundsX()
            double yMin = roi.getBoundsY()
            double w = roi.getBoundsWidth()
            double h = roi.getBoundsHeight()

            int nCols = Math.ceil(w / stepX) as int
            int nRows = Math.ceil(h / stepY) as int

            for (int col = 0; col < nCols; col++) {
                for (int row = 0; row < nRows; row++) {

                    double cx = xMin + col * stepX
                    double cy = yMin + row * stepY
                    if (col % 2 == 1) cy += stepY / 2

                    def pts = []
                    for (int k = 0; k < 6; k++) {
                        double angle = Math.toRadians(60 * k)
                            pts << new Point2(cx + a * Math.cos(angle), cy + a *
Math.sin(angle))
                    }

                    def hexROI = new PolygonROI(pts, plane)
```

```
                    def hexArea = new Area(hexROI.getShape())

                    // Pure mathematical intersection
                    hexArea.intersect(blobArea)

                    if (hexArea.isEmpty()) continue

                    // GEOMETRY ENGINE (GEOMETRY TOOLS)

                    // Convert the Java Area into a Temporary QuPath ROI
                    def tempRoi = RoiTools.getShapeROI(hexArea, plane)

                    // Extract JTS Geometry directly from the ROI
                   def geom = tempRoi.getGeometry()

                            // If the intersection created multiple disconnected
islands, iterate through them.

                    for (int i = 0; i < geom.getNumGeometries(); i++) {
                        def singleGeom = geom.getGeometryN(i)

                           // Re-convert the single island (with its holes) into
the final ROI
                          def finalRoi = GeometryTools.geometryToROI(singleGeom,
plane)

                        // Create the object
                           def obj = PathObjects.createDetectionObject(finalRoi,
blob.getPathClass())
                        allHexes << obj
                    }
                }
            }
        }

        // APPLY CHANGES
        if (!blobsToRemove.isEmpty()) {
            removeObjects(blobsToRemove, true)
            addObjects(allHexes)
        }
    }
}
```

```
/**
 * Function: Calculate Modal Diameter (Global)
 */
double calculateModalDiameter(String channelName, double minAreaThreshold) {
    print "\n--- GLOBAL STATISTICS: ${channelName} ---"
    def pathClass = getPathClass(channelName)
        def detections = getDetectionObjects().findAll { it.getPathClass() == pathClass }

    if (detections.isEmpty()) return 15.0

    def areas = detections.collect { det ->
        return det.getMeasurementList().get("Area µm^2")
    }.findAll { val ->
        val != null && !Double.isNaN(val) && val >= minAreaThreshold
    }

    if (areas.isEmpty()) return 10.0

    double binSize = 5.0
    double maxArea = areas.max()
    int numBins = (int)Math.ceil(maxArea / binSize) + 1
    if (numBins < 1) numBins = 1
    double[] histogram = new double[numBins]

    areas.each { area ->
        int bin = (int)(area / binSize)
        if (bin < numBins) histogram[bin]++
    }

    double[] smoothed = new double[numBins]
    if (numBins > 2) {
            for (int i=1; i<numBins-1; i++) smoothed[i] = (histogram[i-1] + histogram[i] + histogram[i+1]) / 3.0
    } else { smoothed = histogram }

    double maxCount = 0
    int peakIndex = 0
    for (int i=0; i<numBins; i++) {
        if (smoothed[i] > maxCount) { maxCount = smoothed[i]; peakIndex = i }
    }

    double modeArea = peakIndex * binSize + (binSize / 2.0)
```

```
    double exactHexDiameter = 2 * Math.sqrt( (2 * modeArea) / (3 * Math.sqrt(3))
)

      print "     -> Modal Area: ${String.format('%.2f', modeArea)} µm² | Hex
Diameter: ${String.format('%.2f', exactHexDiameter)} µm"

    if (exactHexDiameter < 5.0) return 5.0
    return exactHexDiameter
}

// SCRIPT EXECUTION

// 1. Find Tiles
def tiles = getAnnotationObjects().findAll { it.getPathClass() == null &&
it.getName() != null && it.getName().startsWith("Tile") }
if (tiles.isEmpty()) {
    print "Error: No 'Tile' found."
    return
}
print "Found ${tiles.size()} Tiles. Starting processing..."

// 2. Pre-processing measurements
print "Calculating initial measurements..."
selectDetections()
addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER", "NUCLEUS_CELL_RATIO")

// 3. Calculate optimal diameters
double diametroYellow = calculateModalDiameter("Yellow", 15.0)
double diametroMagenta = calculateModalDiameter("Magenta", 40.0)

// Initialize processors
def processorYellow = new HexGridFromBlobs("Yellow", diametroYellow)
def processorMagenta = new HexGridFromBlobs("Magenta", diametroMagenta)

// 4. LOOP OVER TILES
tiles.eachWithIndex { tile, i ->

    print "Processing Tile ${i+1}/${tiles.size()}..."

    selectObjects(tile)

    processorYellow.runOnParent(tile)
```

```
    processorMagenta.runOnParent(tile)
}

print " Processing completed. Updating hierarchy..."

// 5. FINAL HIERARCHY FIX
resetSelection()

resolveHierarchy()

print "Calculating final measurements..."
selectDetections()
addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER", "NUCLEUS_CELL_RATIO")

// Safe distance calculation
try {
    detectionCentroidDistances()
} catch (Exception e) {
     print "Warning: Could not calculate distances (likely missing one of the
cell classes)."
}

print "Script completed successfully."
```

4.3.3. Algoritmo di fusione spaziale per adiacenza e consolidamento topologico

Al fine di rifinire la segmentazione esagonale e risolvere eventuali artefatti di frammentazione generati ai bordi degli oggetti, è stato sviluppato uno script di post-processing. L'obiettivo dell'algoritmo è aggregare i frammenti minori (small hexes) alle unità principali adiacenti (big hexes), ricostruendo la continuità biologica dell'oggetto rilevato. Il protocollo operativo si articola nelle seguenti fasi:

1. **Stratificazione dimensionale:** L'algoritmo esegue una scansione preliminare di tutti gli oggetti rilevati per un determinato canale (es. "Yellow" o "Magenta"). Gli oggetti vengono classificati in due categorie distinte basandosi su una soglia di area minima critica (minArea = 10.0 µm²):
   - **Elementi principali (big hexes):** Oggetti con superficie maggiore o uguale a 10.0 µm², considerati come il nucleo o il corpo principale della cellula.

- **Frammenti (small hexes):** Oggetti con superficie < 10.0 μm², considerati come frammenti periferici o residui della tassellatura da consolidare.

2. **Indicizzazione spaziale (grid indexing):** Per ottimizzare il carico computazionale e prevenire una complessità di calcolo quadratica (O(N^2)), lo script implementa una struttura dati di hashing spaziale. Lo spazio immagine viene suddiviso virtualmente in una griglia di celle di 50 μm di lato. Ogni oggetto viene assegnato a una cella della griglia in base alle coordinate del suo centroide. Questo permette all'algoritmo di cercare i candidati per la fusione solo nell'immediato vicinato locale, accelerando drasticamente il processo.
3. **Analisi di adiacenza e fusione booleana:** Il cuore del processo è iterativo e si concentra sugli "Elementi Principali":
   - **Buffering:** Per ogni elemento principale, viene generata un'area di influenza estesa (buffer) di 1.0 μm attorno al perimetro originale.
   - **Rilevamento del vicinato:** L'algoritmo interroga la griglia spaziale per recuperare i "Frammenti" candidati situati nelle celle adiacenti.
   - **Test di intersezione:** Viene verificata l'intersezione geometrica tra l'area bufferizzata dell'elemento principale e i frammenti candidati. Se viene rilevata una sovrapposizione (contatto fisico), il frammento viene considerato parte della cellula principale.
   - **Fusione (union):** Utilizzando operazioni di Geometria Costruttiva Solida (CSG), la forma del frammento viene fusa matematicamente con quella dell'elemento principale, creando un nuovo oggetto unico con morfologia complessa. I frammenti assorbiti vengono marcati per evitare doppi conteggi.
4. **Ricostruzione e aggiornamento metrico:** Al termine del ciclo di fusione, gli oggetti originali vengono rimossi dalla gerarchia. Al loro posto vengono generati i nuovi oggetti fusi, per i quali vengono ricalcolati istantaneamente i parametri morfometrici aggiornati (nuova area totale, lunghezza, fattori di forma).

Questo approccio garantisce che i piccoli residui della segmentazione esagonale non vengano scartati come rumore né contati come cellule indipendenti, ma vengano correttamente reintegrati nella biomassa della cellula di appartenenza.

Codice sviluppato:

```
import qupath.lib.objects.PathObjects
import qupath.lib.roi.RoiTools
import java.awt.geom.Area
import java.awt.BasicStroke
import org.slf4j.LoggerFactory

/**
 * HexFusionByAdjacency
 * Merges small hexagonal detections into adjacent larger ones.
 * Adapted to run Tile-by-Tile for better performance and hierarchy management.
 */
class HexFusionByAdjacency {

    String channelName
    double minArea = 10.0
    double gridSize = 50.0
    double buffer = 1.0
    def logger = LoggerFactory.getLogger(HexFusionByAdjacency)

    HexFusionByAdjacency(String channelName) {
        this.channelName = channelName
    }

    Area bufferedArea(def roi, double buffer) {
        def shape = roi.getShape()
        def area = new Area(shape)
        def stroke = new BasicStroke((float)(2 * buffer))
        area.add(new Area(stroke.createStrokedShape(shape)))
        return area
    }

    void runOnParent(def parentObject) {

        // 1. LOAD DETECTIONS FROM TILE
        def allHexes = parentObject.getChildObjects().findAll {
            it.isDetection() && it.getPathClass() == getPathClass(channelName)
        }

        if (allHexes.isEmpty()) return

        // 2. SPLIT BIG / SMALL
            // Note: We assume the area is already calculated. If missing, we
calculate it
```

```
          // on the fly geometrically to avoid errors if measurements are not
updated.

        def bigHexes = []
        def smallHexes = []

        allHexes.each { det ->
            // Try to get the measurement; if missing, use ROI geometry
            def area = det.measurements.get("Area µm^2")
            if (area == null || Double.isNaN(area)) {
                 // Geometric fallback (pixel * calibration)
                 // For simplicity, we assume measurements exist.
                 // If missing, treat as 0.0 (small).
                 area = 0.0
            }

            if (area >= minArea) bigHexes << det
            else smallHexes << det
        }

        if (bigHexes.isEmpty() || smallHexes.isEmpty()) return

        // 3. BUILD SPATIAL GRID
        def grid = [:].withDefault { [] }
        allHexes.each { obj ->
            def roi = obj.getROI()
            def key = [
                Math.floor(roi.getCentroidX() / gridSize),
                Math.floor(roi.getCentroidY() / gridSize)
            ]
            grid[key] << obj
        }

        // 4. FUSION PROCESS
        def newObjects = []
        def absorbedSmalls = [] as Set
        def fusedBigs = [] as Set

        bigHexes.each { big ->
            def roiBig = big.getROI()
            def gx = Math.floor(roiBig.getCentroidX() / gridSize)
            def gy = Math.floor(roiBig.getCentroidY() / gridSize)
```

```
            def candidates = []
            for (dx in -1..1)
                for (dy in -1..1)
                    candidates += grid[[gx + dx, gy + dy]]

            candidates = candidates.findAll { c ->
                def cArea = c.measurements.get("Area µm^2")
                // Fallback area check
                if (cArea == null) cArea = 0.0
                return cArea < minArea && !absorbedSmalls.contains(c)
            }

            def mergedArea = new Area(roiBig.getShape())
            def bufferedBig = bufferedArea(roiBig, buffer)
            def absorbedCount = 0

            candidates.each { small ->
                def a = new Area(bufferedBig)
                a.intersect(new Area(small.getROI().getShape()))
                if (!a.isEmpty()) {
                    mergedArea.add(new Area(small.getROI().getShape()))
                    absorbedSmalls << small
                    absorbedCount++
                }
            }

            if (absorbedCount > 0) {
                                def newROI = RoiTools.getShapeROI(mergedArea,
roiBig.getImagePlane())
                        def newObj = PathObjects.createDetectionObject(newROI,
big.getPathClass())
                newObjects << newObj
                fusedBigs << big
            }
        }

        // 5. APPLY CHANGES
        def objectsToRemove = []
        objectsToRemove.addAll(absorbedSmalls)
        objectsToRemove.addAll(fusedBigs)

        if (!objectsToRemove.isEmpty()) {
            removeObjects(objectsToRemove, true)
```

```
            addObjects(newObjects)
        }
    }
}

// SCRIPT EXECUTION

def tiles = getAnnotationObjects().findAll { it.getPathClass() == null &&
it.getName() != null && it.getName().startsWith("Tile") }

if (tiles.isEmpty()) {
    print "Error: No 'Tile' found."
    return
}

print "Found ${tiles.size()} Tiles. Starting fusion..."

// Initialize Processors
def fuserYellow = new HexFusionByAdjacency("Yellow")
def fuserMagenta = new HexFusionByAdjacency("Magenta")

// LOOP OVER TILES
tiles.eachWithIndex { tile, i ->
    print "--> Processing Tile ${i+1} / ${tiles.size()} (${tile.getName()})"

     // Execute Fusion directly on the tile (without selecting it to avoid log
spam)
    fuserYellow.runOnParent(tile)
    fuserMagenta.runOnParent(tile)
}

print " Tile processing completed."

// HIERARCHY UPDATE
resetSelection()
resolveHierarchy()

print "Calculating final measurements..."
selectDetections()
detectionCentroidDistances()
// This recalculates measurements for the newly fused objects
addShapeMeasurements("AREA", "LENGTH", "CIRCULARITY", "SOLIDITY",
"MAX_DIAMETER", "MIN_DIAMETER", "NUCLEUS_CELL_RATIO")
```

```
print "Fusion Script completed successfully."
```

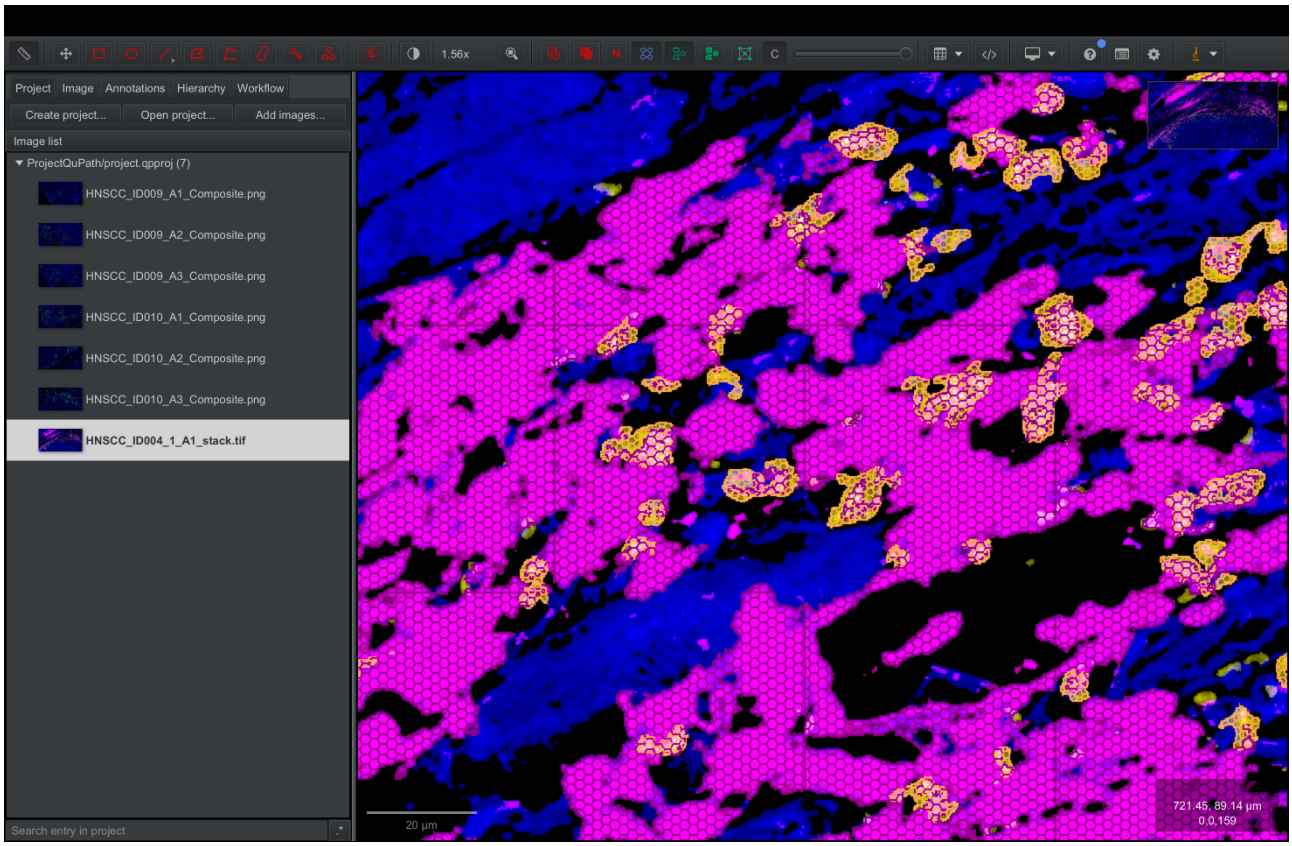

Figura 4.3 - Segmentazione manuale

5. Script R per analisi statistiche

A conclusione della fase di segmentazione e fenotipizzazione su *QuPath*, il dataset quantitativo ottenuto è stato sottoposto ad analisi statistica. Preliminarmente alla descrizione dello script specifico elaborato per questo studio, è opportuno definire l'architettura software utilizzata, basata sul linguaggio *R* e sull'ambiente di sviluppo *RStudio*.

5.1. R e RStudio

Per l'elaborazione dei dati quantitativi derivanti dall'analisi d'immagine e per la successiva analisi statistica, questo studio si è avvalso del linguaggio di programmazione *R*, gestito attraverso l'ambiente di sviluppo integrato (IDE) *RStudio*.

R: Il motore computazionale

R è un linguaggio di programmazione e un ambiente software libero (open source) dedicato al calcolo statistico e alla produzione di grafici. Nato come evoluzione del linguaggio "*S*" presso i Bell Laboratories, *R* è oggi lo standard di riferimento nella ricerca accademica e bioinformatica grazie alla sua flessibilità e alla vasta disponibilità di algoritmi statistici avanzati. Nel contesto di questa tesi, *R* funge da "motore" computazionale: esegue i calcoli, manipola i dataset ed elabora i modelli statistici.

RStudio: L'interfaccia operativa

Sebbene *R* possa essere utilizzato tramite una riga di comando essenziale, l'analisi è stata condotta all'interno di *RStudio*, un'interfaccia grafica (IDE) progettata per rendere l'uso di *R* più accessibile, organizzato e produttivo. L'architettura di *RStudio* suddivide lo spazio di lavoro in quattro quadranti funzionali che permettono il controllo simultaneo di tutte le fasi dell'analisi:

- **Source pane (editor degli script):** È l'area dedicata alla scrittura e al salvataggio del codice. L'utilizzo di script è fondamentale per la ricerca scientifica in quanto garantisce la riproducibilità dell'analisi: ogni passaggio, dalla pulizia dei dati grezzi alla generazione dei grafici finali, viene registrato e può essere rieseguito o verificato in qualsiasi momento.
- **Console:** Rappresenta l'interfaccia diretta con il motore *R*. Qui vengono eseguiti i comandi e visualizzati gli output testuali immediati (es. risultati di test statistici).
- **Environment & history:** Questo pannello permette di monitorare in tempo reale gli "oggetti" caricati nella memoria di lavoro (es. i dataframe contenenti le misurazioni cellulari importate da *QuPath*), visualizzandone struttura e dimensioni.
- **Files, plots, packages & help:** Un'area polifunzionale che consente di navigare nelle directory del progetto, visualizzare ed esportare i grafici generati (come boxplot o scatterplot), gestire le librerie aggiuntive e consultare la documentazione tecnica delle funzioni.

Estensibilità tramite pacchetti (packages)

Un punto di forza cruciale dell'ecosistema R/RStudio è la sua struttura modulare. Le funzionalità di base sono state estese tramite l'installazione di pacchetti specifici scaricati dal repository ufficiale CRAN (Comprehensive R Archive Network). Questi pacchetti (ad esempio *ggplot2* per la visualizzazione avanzata o *dplyr* per la manipolazione dei dati) forniscono raccolte di funzioni, dati e codice compilato che hanno permesso di adattare l'ambiente di lavoro alle specifiche esigenze dell'analisi del microambiente tumorale [10].

5.1.1. Procedura di aggregazione e calcolo delle medie spaziali in ambiente R

Al fine di ottenere indicatori rappresentativi per ciascuna regione di tessuto analizzata, i dati grezzi derivanti dalla segmentazione d'immagine sono stati sottoposti a una procedura di elaborazione automatizzata in ambiente statistico. Il flusso di lavoro si è articolato nelle seguenti fasi sequenziali:

1. **Acquisizione e lettura del dataset:** La procedura ha inizio con l'importazione del file contenente le misurazioni spaziali complete. Il sistema è configurato per permettere all'operatore di selezionare manualmente il dataset di input, garantendo la corretta interpretazione della codifica dei caratteri e della struttura tabulare originale.
2. **Raggruppamento e calcolo delle medie spaziali:** Il nucleo dell'elaborazione consiste nella sintetizzazione dei dati puntuali (riferiti alle singole cellule) in dati aggregati per unità spaziale (riferiti ai "Tiles" o regioni di interesse). L'algoritmo raggruppa le misurazioni in base all'area di appartenenza e calcola la media aritmetica delle distanze reciproche tra le diverse popolazioni linfocitarie, fornendo un valore unico di densità o prossimità per ogni regione campionata.
3. **Analisi specifica delle sottopopolazioni e strutture linfoidi:** Parallelamente all'analisi globale, viene isolato un sottoinsieme di dati limitato esclusivamente alla popolazione B-linfocitaria. Su questo specifico campione, l'algoritmo calcola la distanza media (con segno) che separa tali cellule dai margini delle Strutture Terziarie Linfoidi (TLS). Il calcolo viene differenziato in base al grado di maturazione delle strutture, generando metriche distinte per le TLS mature e immature.
4. **Integrazione e consolidamento del dataset:** I risultati parziali derivanti dalle diverse operazioni di calcolo (distanze inter-cellulari e distanze dalle strutture TLS) vengono successivamente unificati. Attraverso un'operazione di fusione logica basata sull'identificativo univoco della regione, le diverse metriche vengono allineate in un unico prospetto riassuntivo, garantendo che ogni unità di analisi disponga del set completo di parametri calcolati.
5. **Esportazione dei risultati:** Nella fase conclusiva, il dataset finale consolidato viene esportato e salvato in un formato standard compatibile con i successivi software di analisi statistica. Il file di output viene archiviato automaticamente nella medesima directory dei dati di origine, assicurando l'ordine e la tracciabilità del flusso di lavoro.

Codice sviluppato:

```
# R SCRIPT FOR SPATIAL STATISTICAL ANALYSIS
# Objective: Aggregate spatial measurements per Region of Interest (ROI/Tile)
```

```
# 1. DATA IMPORT
# Open a system dialog to manually select the input CSV file containing raw
measurements
file_path <- file.choose()

# Read the dataset specifying the separator (;) and encoding
dati <- read.csv(file_path, sep = ";", fileEncoding = "UTF-8")

# Print column names to console for structure verification
print(names(dati))



# 2. GLOBAL SPATIAL AGGREGATION
# Compute the mean distance to specific cell populations for each ROI (Parent).
# Grouping variable: "Parent" (identifies the Tile or Region)

# Mean distance to 'Green' detections (e.g., CD3+ T-cells)
media_distanza_green <- aggregate(Distance.to.detection.with.Green.µm ~ Parent,
data = dati, FUN = mean)

# Mean distance to 'Red' detections (e.g., CD20+ B-cells)
media_distanza_red <- aggregate(Distance.to.detection.with.Red.µm ~ Parent, data
= dati, FUN = mean)



# 3. TLS INFILTRATION ANALYSIS
# Analyze the proximity of specific populations to Tertiary Lymphoid Structures
(TLS).

# Subset the dataset to isolate only the 'Red' population (Target Phenotype)
dati_red <- subset(dati, Classification == "Red")

# Compute mean signed distance from 'Red' cells to Mature TLS annotations
media_distanza_TLSmatura <-
aggregate(Signed.distance.to.annotation.with.TLSmatura.µm ~ Parent, data =
dati_red, FUN = mean)

# Compute mean signed distance from 'Red' cells to Immature TLS annotations
media_distanza_TLSimmatura <-
aggregate(Signed.distance.to.annotation.with.TLSimmatura.µm ~ Parent, data =
dati_red, FUN = mean)

# Display partial results for verification
```

```
print(media_distanza_green)


# 4. DATA MERGING
# Load necessary libraries for data manipulation
library(dplyr)
library(purrr)

# Create a list containing all the partial dataframes generated above
liste_medie <- list(media_distanza_green, media_distanza_red,
media_distanza_TLSmatura, media_distanza_TLSimmatura)
print(liste_medie)

# Merge all dataframes into a single summary table using a full outer join on
"Parent"
# 'reduce' iteratively applies the join function to the list elements
distanze_combine <- reduce(liste_medie, full_join, by = "Parent")

# Display the final combined dataframe
print(distanze_combine)


# 5. DATA EXPORT
# Define the output path: saves the file in the same directory as the input file
save_path <- file.path(dirname(file_path), "distanze_medie_combinate.csv")

# Write the final table to a CSV file
# Format: semicolon separator, dot for decimals, no row names
write.table(distanze_combine, save_path, row.names = FALSE, sep = ";", dec =
".")

# Print confirmation message
message("File successfully saved to: ", save_path)
```

6. Risultati sperimentali

Il presente studio ha preso in esame un totale di 38 campioni di tessuto fissato in formalina e incluso in paraffina (FFPE), derivati da una coorte di 21 pazienti con diagnosi confermata di neoplasia. L'analisi quantitativa è stata condotta su un totale di 94 ROI. Al fine di garantire la rilevanza clinica e la correttezza istologica delle aree indagate, la selezione e la delimitazione di tutte le ROI sono state sottoposte a supervisione e validazione diretta da parte dell'anatomo-patologo di riferimento.

Preliminarmente all'applicazione dell'algoritmo di tassellatura esagonale (post-processing), è stato condotto un esperimento di validazione tecnica per valutare l'efficacia del protocollo di segmentazione automatica basato sui *Pixel Classifiers*. Poiché la successiva suddivisione geometrica opera esclusivamente sugli oggetti (blobs) già identificati, eventuali cellule non rilevate in questa fase iniziale costituirebbero dei "falsi negativi" permanenti, esclusi definitivamente dall'analisi spaziale. Era pertanto fondamentale quantificare la perdita di informazione iniziale (errore di rilevamento).

Per la stima dell'errore è stato selezionato un sottoinsieme rappresentativo di 3 ROI. All'interno di ciascuna regione, l'analisi è stata focalizzata su un singolo "Tile" (unità di campionamento 510x510 μm), scelto intenzionalmente per le sue caratteristiche di bassa densità cellulare e ridotta presenza di aggregati complessi. La scelta di aree a bassa densità è stata metodologicamente necessaria per stabilire un ground truth affidabile: in assenza di cluster sovrapposti, infatti, la discriminazione delle singole unità cellulari da parte dell'occhio umano è inequivocabile e priva di ambiguità. Per ogni Tile selezionato si è proceduto al confronto tra:

1. **Conteggio manuale:** Il numero di cellule positive identificate visivamente dall'operatore.
2. **Conteggio automatico:** Il numero di cellule rilevate dallo script automatizzato.

L'accuratezza del sistema è stata valutata calcolando l'errore percentuale medio rispetto al conteggio manuale. L'analisi ha restituito un valore di errore pari al 7,89%. Tale scostamento, attribuibile principalmente alla mancata rilevazione di elementi con intensità di segnale al limite della soglia di background (faint staining), è stato giudicato dell'anatomo-patologo ampiamente entro i limiti di

tolleranza per l'analisi di tessuti eterogenei come quelli neoplastici, confermando l'idoneità dell'algoritmo di segmentazione prima della fase di raffinamento esagonale.

Campione Validazione	Tile ID	Conteggio Manuale (Ground Truth)	Conteggio Automatico (Script)	Errore Relativo (%)
HNSCC_ID009_A1	2	*38*	*37*	2,63
HNSCC_ID009_A2	6	*24*	*21*	12,50
HNSCC_ID009_A3	6	*14*	*12*	14,28
Media Totale				**7,89**

Tabella 1 - Analisi dell'errore di segmentazione preliminare

7. Conclusioni e sviluppi futuri

Il presente lavoro di Tesi si è posto l'obiettivo di sviluppare e validare un flusso di lavoro computazionale integrato per l'analisi quantitativa del microambiente tumorale in campioni di carcinoma a cellule squamose del distretto testa-collo. L'approccio metodologico adottato, basato sull'utilizzo della piattaforma open-source *QuPath* e sull'implementazione di script personalizzati in linguaggio *Groovy*, ha permesso di superare le limitazioni intrinseche degli algoritmi di segmentazione standard, in particolare nella gestione delle aree ad alta densità cellulare.

I risultati ottenuti evidenziano come l'introduzione della tassellatura esagonale adattiva costituisca una soluzione efficace per risolvere il problema della sotto-segmentazione negli aggregati confluenti (blobs). A differenza dei metodi tradizionali che tendono a scartare o sottostimare tali regioni, il protocollo sviluppato ha permesso il recupero sistematico dell'informazione spaziale, garantendo una mappatura continua e dettagliata della distribuzione cellulare. La validazione tecnica, condotta attraverso il confronto diretto con il ground truth manuale, ha confermato l'affidabilità dell'algoritmo sviluppato, con un tasso di errore contenuto e compatibile con le esigenze della ricerca clinica. Ciò ha reso possibile l'estrazione di metriche spaziali robuste riguardanti le popolazioni immunitarie chiave (CD3, CD20, CD8, CD163), fornendo dati inediti sulla topografia dell'infiltrazione linfocitaria e sulla relazione spaziale con le strutture terziarie linfoidi.

In sintesi, lo studio conferma che l'integrazione tra istopatologia digitale e analisi computazionale avanzata non solo migliora la precisione del conteggio cellulare, ma apre nuove prospettive per la fenotipizzazione spaziale del tumore, offrendo strumenti quantitativi potenzialmente predittivi per la stratificazione prognostica del paziente.

Sebbene l'approccio di tassellatura esagonale sviluppato abbia dimostrato un'elevata efficacia nel risolvere le problematiche di sotto-segmentazione, sono state identificate specifiche aree di miglioramento algoritmico che costituiranno l'oggetto di future implementazioni. In primo luogo, si prevede di raffinare il metodo di stima della dimensione cellulare, parametro critico per la generazione della griglia. Attualmente, la dimensione degli esagoni è determinata statisticamente

sulla base della moda della distribuzione delle aree rilevate. Tuttavia, in contesti neoplastici caratterizzati da forte eterogeneità (anisocitosi) o dalla presenza simultanea di popolazioni cellulari dimensionalmente distinte (*e.g*. piccoli linfociti *vs* grandi cellule tumorali), l'uso di una singola moda potrebbe risultare limitante. Un'evoluzione naturale dello script consisterà nell'abbandonare la stima unimodale a favore di modelli statistici più complessi, come le “Misture di Gaussiane” (Gaussian Mixture Models), capaci di identificare multiple sottopopolazioni e adattare dinamicamente la scala della tassellatura alle diverse componenti del tessuto. In secondo luogo, si intende superare la rigidità geometrica della griglia attuale integrando un'analisi basata sul gradiente dell'immagine. L'attuale tassellatura esagonale opera in modo "cieco" rispetto al contenuto locale del blob; l'obiettivo futuro è trasformare tale griglia in una mesh adattiva. Applicando algoritmi di deformazione guidati dai vettori di gradiente (o varianti dell'algoritmo *Watershed* vincolato alla griglia), i bordi degli esagoni potranno "rilassarsi" e adattarsi ai minimi locali di intensità, allineandosi così ai reali confini biologici (membrane o bordi nucleari). Questo approccio ibrido permetterebbe di coniugare la regolarità spaziale della tassellatura con la fedeltà morfologica della segmentazione classica.

L'implementazione completa degli script utilizzati nel presente progetto è disponibile pubblicamente presso il seguente *repository* GitHub: https://github.com/AmbroAle/Tesi-Irst.git

Bibliografia

[1] Argiris A, Karamouzis MV, Raben D, Ferris RL. Head and neck cancer. Lancet. 2008 May 17;371(9625):1695-709. doi: 10.1016/S0140-6736(08)60728-X. PMID: 18486742; PMCID: PMC7720415.

[2] Zou J, Zhang Y, Zeng Y, Peng Y, Liu J, Xiao C, Wu F. Tertiary Lymphoid Structures: A Potential Biomarker for Anti-Cancer Therapy. Cancers (Basel). 2022 Dec 2;14(23):5968. doi: 10.3390/cancers14235968. PMID: 36497450; PMCID: PMC9739898.

[3] Marchetti, M. Segmentazione *automatica di regioni in immagini istologiche* (Doctoral dissertation)

[4] Ajileye, A. e Esan, E. 2022. Fissazione e fissativi in istopatologia: una revisione. *Bayero Journal delle scienze pure e applicate* . 15, 1 (dicembre 2022), 231–243. DOI: https://doi.org/10.4314/bajopas.v15i1.32.

[5] Cappi G, Dupouy DG, Comino MA, Ciftlik AT. Ultra-fast and automated immunohistofluorescent multistaining using a microfluidic tissue processor. Sci Rep. 2019 Mar 14;9(1):4489. doi: 10.1038/s41598-019-41119-y. PMID: 30872751; PMCID: PMC6418167.

[6] Bankhead P, Loughrey MB, Fernández JA, Dombrowski Y, McArt DG, Dunne PD, McQuaid S, Gray RT, Murray LJ, Coleman HG, James JA, Salto-Tellez M, Hamilton PW. QuPath: Open source software for digital pathology image analysis. Sci Rep. 2017 Dec 4;7(1):16878. doi: 10.1038/s41598-017-17204-5. PMID: 29203879; PMCID: PMC5715110.

[7] Schindelin J, Arganda-Carreras I, Frise E, Kaynig V, Longair M, Pietzsch T, Preibisch S, Rueden C, Saalfeld S, Schmid B, Tinevez JY, White DJ, Hartenstein V, Eliceiri K, Tomancak P, Cardona A. Fiji: an open-source platform for biological-image analysis. Nat Methods. 2012 Jun 28;9(7):676-82. doi: 10.1038/nmeth.2019. PMID: 22743772; PMCID: PMC3855844.

[8] Stringer C, Wang T, Michaelos M, Pachitariu M. Cellpose: a generalist algorithm for cellular segmentation. Nat Methods. 2021 Jan;18(1):100-106. doi: 10.1038/s41592-020-01018-x. Epub 2020 Dec 14. PMID: 33318659.

[9] Kornilov A, Safonov I, Yakimchuk I. A Review of Watershed Implementations for Segmentation of Volumetric Images. J Imaging. 2022 Apr 26;8(5):127. doi: 10.3390/jimaging8050127. PMID: 35621890; PMCID: PMC9146301.

[10] Hair, Joseph & Hult, G. Tomas M. & Ringle, Christian & Sarstedt, Marko & Danks, Nicholas & Ray, Soumya. (2021). Overview of R and RStudio. 10.1007/978-3-030-80519-7_2.

Ringraziamenti

Desidero esprimere la mia più sincera gratitudine al mio relatore, la Prof.ssa Antonella Carbonaro dell'Università di Bologna, per avermi offerto l'opportunità di lavorare a questa Tesi, permettendomi così di dare il mio contributo nell'ambito della ricerca sperimentale oncologica.

Un sentito ringraziamento va alle mie correlatrici: la Dott.ssa Marcella Tazzari, la Dott.ssa Maria Maddalena Tumedei dell'IRCCS IRST di Meldola e la Dott.ssa Rebecca Polidori dell' Università degli Studi di Milano. La loro disponibilità, gentilezza e competenza tecnica durante tutte le fasi del progetto sono state per me un punto di riferimento fondamentale.

Un ringraziamento speciale è rivolto al Prof. Filippo Piccinini, per avermi coinvolto attivamente in questo lavoro e per essere stato una guida costante e presente. Grazie per il confronto settimanale, per la pazienza nel chiarire ogni mio dubbio e per il prezioso supporto fornito durante la stesura di questo elaborato.

Ringrazio di cuore i ragazzi della "Polivalente", con i quali ho condiviso momenti di studio e di svago. Il confronto continuo e le risate scambiate hanno reso questo percorso più piacevole, permettendomi di affrontare le sfide universitarie con maggiore serenità.

Infine, il ringraziamento più importante va alla mia famiglia, che mi ha sostenuto incondizionatamente nella scelta di riprendere gli studi. Grazie per avermi supportato, incoraggiato e per aver creduto in me in ogni momento di questo cammino.