UNIVERSITÀ DEGLI STUDI DI BARI

ALDO MORO

Dipartimento di Informatica

Corso di Laurea in Informatica e Tecnologie per la Produzione del Software

---

TESI DI LAUREA

IN

SISTEMI COOPERATIVI

SISTEMA SOFTWARE BASATO SU PATTERN RECOGNITION PER L'ESTRAZIONE DI FEATURES DA CELLULE LEUCOCITARIE

Relatore:

Chiar.mo Prof. Giovanni Dimauro

Correlatore:

Chiar.mo Dott. Francesco Girardi

Laureando:

Michele Cafagna

Anno Accademico 2016/2017

Abstract

Nel lavoro di tesi, sono stati studiati ed implementati differenti algoritmi per la segmentazione e l'estrazione di un leucocita, da una regione circoscritta di un'immagine di uno striscio di sangue. Tali algoritmi sono stati confrontati e valutati e viene perciò fornita, la migliore soluzione trovata, basata su sogliatura di Otsu, su tecniche di eliminazione del background e su operazioni morfologiche. Inoltre viene presentata una tecnica per il conteggio automatico dei lobi nucleari basata su una rappresentazione a grafo dello scheletro del nucleo. I lobi, vengono suddivisi dal nucleo, attraverso l'analisi delle proprietà dello scheletro che verrà poi decomposto in segmenti. Tutte le tecniche proposte sono state sperimentate e perfezionate, utilizzando un dataset di una decina di migliaia di cellule. Verranno descritti vantaggi e limiti di tali algoritmi, confrontati con algoritmi presenti in letteratura.

INDICE

1 PROBLEMA CLINICO

1.1 Analisi morfologica su striscio di sangue, per la diagnosi di patologie legate ad anomalie di cellule leucocitarie

Il conteggio differenziale dei leucociti costituisce un importante elemento di valutazione diagnostica per il trattamento di molte malattie, di terapie in corso o di screening e check-up; è un'operazione manuale, faticosa e richiede molto tempo.

Durante la conta vengono effettuate anche altre operazioni [8][5]:

- Segnalazione di alterazione di forma
- Anomalie peculiari di patologia
- Descrizione di anomalie qualitative e quantitative.

Nel caso di disturbi qualitativi abbiamo tutte quelle anomalie che interessano le funzionalità e l'aspetto dei leucociti, ereditarie o acquisite, croniche o temporanee, tra cui:

- **Ipersegmentazione**: può essere osservata in corso di anemia megaloblastica o in altre condizioni in cui è danneggiata la sintesi del DNA come nel caso di malattia di radiazioni o nell'assunzione di alcuni farmaci. Si manifesta come un aumento del normale numero di lobi che normalmente varia da 2 a 5.
- **Iposegmentazione**: il nucleo non è lobato è si presenta sotto forma ellittica e a fagiolo. La funzione dei leucociti rimane normale, ma possono verificarsi delle violazioni acquisite nelle infezioni da leucemia.
- **Carenza di mieloperossidasi**: mieloperossidasi è un enzima presente nei granuli di neutrofili e monociti che genera specie reattive dell'ossigeno per distruggere particelle estranee. Causa problemi se coincide con il diabete mellito, altrimenti non ha particolari conseguenze
- **Sindrome di May-Hegglin**: si tratta di un'anomalia autosomica dominante, caratterizzata da piastrine grandi. Il sintomo principale è la presenza dei cosiddetti corpi di Dhole (vistose aree grigio-blu) nel citoplasma dei leucociti
- **Sindrome di Chediak-Higashi**: patologia rara che porta alla morte nel giro di un mese, se non si interviene con trapianto di midollo osseo. I granulociti mostrano granuli di dimensioni enormi e le funzioni di chemiotassi e fagocitosi sono compromesse
- **Granulazioni tossiche**: I granuli dei neutrofili mostrano un colorito più corposo, tendente al blu, come risultato di una coagulazione proteica in infezioni gravi.

Numerose condizioni patologiche possono determinare alterazioni quantitative. Si definisce **leucocitosi** l'aumento del numero di leucociti nel sangue oltre i valori normali, mentre la **leucopenia**, si manifesta come l'esatto opposto. Se l'incremento o la riduzione non interessano tutte le serie, si utilizzano a seconda della serie interessata i termini di neutrofilia, eosinofilia, basofilia, linfocitosi, monocitosi.
Con il termine leucemia, si indicano le patologie neoplastiche ematologiche croniche e acute, linfoidi e mieloidi, che originano dalla trasformazione neoplastica di una cellula ematica del midollo osseo, con una proliferazione incontrollata della cellula trasformata e conseguente immissione nel circolo ematico di cellule immature (blasti) quantitativamente in eccesso e qualitativamente diverse dalle cellule mature. La loro presenza inibisce il regolare funzionamento del midollo osseo.
Finora, la conta dei leucociti è stata effettuata osservando al microscopio ottico un campione del sangue prelevato, strisciato sul vetrino e colorato, così da ottenere una stima attendibile su un campione di circa 200 cellule, determinando la cosiddetta **formula leucocitaria**.
Per formula leucocitaria (o emogramma) si intende la determinazione percentuale dei vari tipi cellulari di leucociti presenti nello striscio di sangue periferico.

Normalmente la formula è la seguente [13]:

- Neutrofili: tra 50 e 70%
- Eosinofili: tra 2 e 3%
- Basofili: tra 0.1 e 1%
- Linfociti: tra 20 e 30%
- Monociti: tra 3 e 8%

Questi valori rappresentano una stima orientativa e suscettibile a frequenti fluttuazioni.

2 LEUCOCITI

I leucociti sono cellule del sangue. La loro funzione principale è quella di preservare l'integrità biologica dell'organismo tramite l'attuazione di meccanismi di difesa. Sono coinvolti nella difesa dell'organismo, contro le aggressioni di agenti esterni, quali batteri, miceti, virus ed i loro prodotti (tossine)

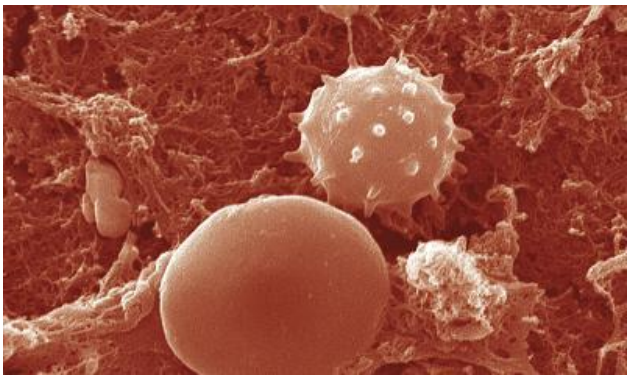

Figura 2.1Macrofotografia ottenuta al microscopio elettronico con un eritrocita (in basso) e un leucocita (in alto).

Si presentano incolori e traslucidi, in quanto sprovvisti di pigmento si suddividono sulla base delle loro caratteristiche morfofunzionali in due macrocategorie: granulociti e agranulociti [13]. I granulociti devono il loro nome alla presenza di granulazioni citoplasmatiche e hanno un nucleo lobato, sono a loro volta distinguibili in eosinofili, basofili e neutrofili, mentre al gruppo degli agranulociti a cui appartengono monociti e linfociti, sono caratterizzati dall'assenza di granuli e possiedono nuclei non lobati. I leucociti svolgono un ruolo chiave nell'efficienza del sistema immunitario del corpo umano, proteggendolo da virus, agenti batterici e da tutti gli altri organismi invasori a cui siamo esposti, interagendo con il mondo che ci circonda. Hanno la capacità di inglobare le cellule morte, i germi e i batteri mediante fagocitosi, rilasciando inoltre speciali proteine antimicrobiche e producono gli anticorpi. I globuli bianchi oltre ad essere presenti nel flusso sanguigno, possono grazie alla loro mobilità uscire dal circolo e spostarsi all'interno dei tessuti infiammatori, dove si svolge la parte più delicata della loro azione. Nell'uomo adulto e in salute sono presenti tra i 4000 e gli 11000 leucociti ogni microlitro di sangue, ma si tratta di numeri soggetti a molte fluttuazioni. Anche il solo passaggio dalla veglia al sonno può ridurre il numero di diverse unità, mentre un'infezione

batterica ne aumenta la proliferazione. Ci sono cinque tipologie di globuli bianchi nel sangue, con compiti differenti, e con specifiche proprietà, ne seguono le caratteristiche principali di ognuno di essi.

2.1 Neutrofili

"I neutrofili costituiscono la componente cellulare quantitativamente più rilevante dei globuli bianchi (50-70%)" [13] (S. Adamo, 2002, 567). Sono elementi mobili e hanno capacità fagocitaria. Svolgono un ruolo difensivo contro infezioni batteriche e la loro funzione è legata a quella dei linfociti.

Figura 2.2 Un neutrofilo con evidente nucleo lobato.

Presentano una forma globulosa ed un diametro di 10-12 μm. Il nucleo è polisegmentato con un numero di lobi compreso fra 2 e 5, uniti da ponti di materiale nucleico. Il numero di lobi cresce con l'età della cellula. Nel citoplasma si osservano granulazioni di colore rosso-lilla, suddivisibili in base alla loro funzione. L'età è stimabile usando la *fomula di Arneth* [13]*:*

- Nucleo a ferro di cavallo (5%)
- Segmentato con due lobuli (25%)
- Segmentato con tre lobuli (40%)
- Segmentato con quattro lobuli (25%)
- Segmentato con cinque lobuli (5%)

2.2 Linfociti

"I linfociti costituiscono la componente cellulare fondamentale del sistema immunitario (immunità acquisita) provvedendo al riconoscimento specifico degli antigeni esterni e svolgendo le principali funzioni effettrici." [13] (S. Adamo, 2002, 576)

I linfociti hanno forma rotondeggiante, e si distinguono per le dimensioni in *piccoli linfociti* (diametro di 6-9 μm) e in *grandi linfociti* (diametro di 9-15 μm). I piccoli linfociti sono la componente più numerosa, con un grande nucleo ovale o reniforme e cromatina addensata, il citoplasma circonda il nucleo con un leggero alone debolmente basofilo e rare granulazioni. I grandi linfociti (circa

10%) presentano un nucleo con meno cromatina addensata, un nucleolo, citoplasma abbondante e granulazioni azzurrofile.

I linfociti rappresentano il 20-30% dei globuli bianchi presenti nel sangue circolante; sono presenti anche nella linfa, nei linfonodi e nei tessuti connettivi, sono in grado di uscire dal circolo sanguigno e rientrarvi sfruttando le vie linfatiche, non svolgono attività fagocitaria, posseggono una mobilità. I linfociti circolanti, sono caratterizzati dalla presenza di antigeni di membrana che consentono di caratterizzare diverse sottopopolazioni cellulari che sono rappresentate da linfociti B e dai linfociti T, questi ultimi a loro volta suddivisi in T-citotossici e T-helper. L'essenziale differenza tra i linfociti B e T è basata sulla forma dell'antigene che sono in grado di riconoscere, entrambi sono coinvolti nei meccanismi di difesa immunitaria acquisita. I linfociti T sono in grado di uccidere cellule infettate da virus, e di stimolare i linfociti B nella produzione di anticorpi, ed è anche la loro principale funzione. Una categoria particolare, sono i linfociti NK (natural killer), importanti cellule effettrici della risposta immunitaria innata. Quest'ultimi oltre alla grande dimensione e al nucleo indentato, possiedono enzimi litici, che sono utilizzati per l'eliminazione di cellule modificate per infezione o malformazioni e che risultano prive di antigeni di istocompatibilità (self), partecipano anche alla risposta immuno-specifica [13].

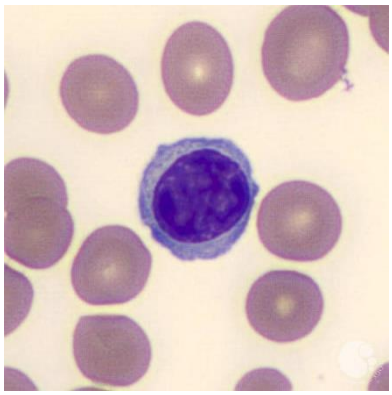

Figura 2.3 Un linfocita, il suo nucleo comprende gran parte della cellula.

2.3 Monociti

I monociti sono gli elementi corpuscolati del sangue appartenenti al sistema monocito-macrofago. Tale sistema interviene nei meccanismi antibatterici, nella risposta infiammatoria. Il monocito risponde agli stimoli infiammatori e chemiotattici raggiungendo la sede dell'infiammazione dove aumentano di dimensioni e si arricchiscono di lisosomi maturando in macrofagi. I monociti sono in grado di digerire un gran numero di batteri, o cellule morte e particelle indesiderate. Non sono in grado di riconoscere autonomamente una minaccia,

piuttosto attaccano soltanto dopo che gli anticorpi si legano al corpo estraneo evidenziandone la pericolosità.

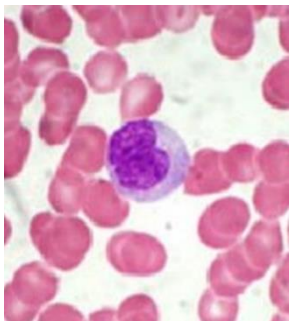

Figura 2.4 Un monocita con il tipico nucleo a fagiolo.

Il monocito si presenta rotondeggiante, voluminoso (12-18 μm di diametro) e rappresenta il 3-8% della popolazione leucocitaria [13]. Il nucleo, situato eccentricamente, è generalmente reniforme, con la cromatina disposta in aggregati tra loro connessi da filamenti sottili. Il citoplasma ha una colorazione blu-grigiastra con granuli color porpora.

2.4 Eosinofili

Gli eosinofili sono dotati di un'attività selettiva nei confronti delle infezioni parassitarie. Meno dell'1% del numero totale degli eosinofili circola nel sangue, la maggior parte si trova nel midollo osseo e nei tessuti. In genere il tempo di sopravvivenza in coltura di un eosinofilo è superiore a quello del neutrofilo (8-12 giorni contro 2-4 giorni). Vengono attirati nelle regioni di interazione antigene/anticorpo per il rilascio di un fattore chemiotattico eosinofilo, contenuto nei granuli e provvedono alla fagocitosi del complesso antigene/anticorpo

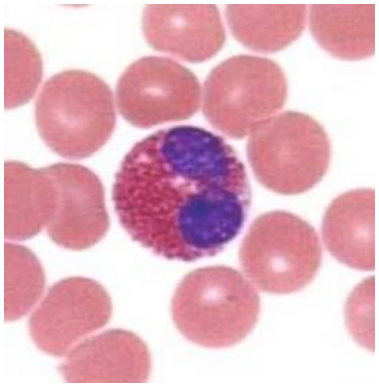

Figura 2.5 Un eosinofilo con evidente colorazione rossastra e nucleo bilobato.

Hanno un diametro di circa 12 μm e rappresentano il 2-4% dei leucociti, il nucleo è generalmente bilobato, con lobi collegati da un sottile filamento di cromatina. Il citoplasma è caratterizzato dalla presenza di granuli specifici, ovoidali, numerosi e di diametro uniforme (circa 0.5 μm), acidofili (colorabili in giallo-arancio con l'eosina) [13]. All'interno di questi granuli, sono presenti sostanze capaci di mediare e regolare le reazioni difensive nelle quali sono coinvolte.

2.5 Basofili

I basofili sono responsabili della ipersensibilità immediata (quali riniti, allergie orticaria, asma, anafilassi). Sono inoltre implicati nella patogenesi della ipersensibilità cutanea ritardata e nella risposta tardiva dell'anafilassi. I basofili sono in grado inoltre di liberare altre sostanze (dai loro granuli) che hanno ruolo nei processi infiammatori, la liberazione di queste sostanze è responsabile dell'ipersensibilità immediata. Possiedono anticorpi coinvolti soprattutto nelle reazioni allergiche, e possono essere attivati dal legame con lo specifico antigene.

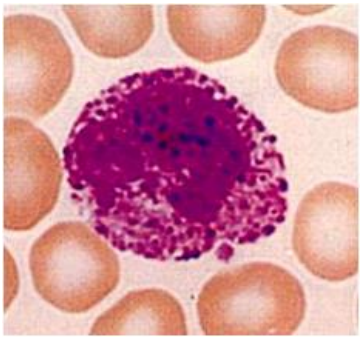

Figura 2.6 Un basofilo il nucleo è quasi coperto dai granuli cellulari.

Hanno in media un diametro di circa 10 μm e sono presenti nel sangue circolante in una concentrazione pari allo 0.5-1% dei leucociti [13]. Il nucleo è bilobato con cromatina addensata. Nel citoplasma sono presenti voluminose granulazioni, che impartiscono una intensa colorazione basofila e metacromatica.

3 OBIETTIVO

L'obiettivo generale è dunque quello di sviluppare un sistema di riconoscimento cellulare, in grado di classificare automaticamente le tipologie di leucociti, in modo da assistere il lavoro di segnalazione e analisi dello specialista fornendo in anticipo l'identificazione e classificazione del leucocita e la rilevazione di eventuali anomalie, potendo in tal modo costituire un'efficiente supporto per un primo screening di patologiche ematologiche ed internistiche.

Ci sono lavori in letteratura, in cui è stato analizzato il problema della conta automatica dei leucociti ed è stato possibile realizzare diversi algoritmi di classificazione utilizzando differenti tecniche riguardanti l'elaborazione delle immagini. Nella maggior parte di essi si può evidenziare la separazione dell'algoritmo in tre fasi: identificazione dei globuli bianchi, estrazione delle features e relativa classificazione. Nella prima fase, sono state utilizzate tecniche riguardanti la segmentazione basata su colore o la trasformata di watershed, edge detection e di clustering, nella la seconda vengono impiegate tecniche basate su operazioni morfologiche e analisi del colore, per la terza invece, classificatori bayesiani, k-nn, reti neurali, e tecniche di analisi delle componenti principali."

Il mio lavoro di tesi è incentrato sulla seconda parte del sistema relativa all'identificazione della cellula ed estrazione delle features. Questo lavoro non è semplice implementazione di algoritmi, ma è incentrata sulla sperimentazione, in parte basata su tecniche presenti in letteratura, e in parte su evidenze sperimentali, condotte personalmente su un dataset creato ex novo.

4 ARCHITETTURA GENERALE DEL SISTEMA

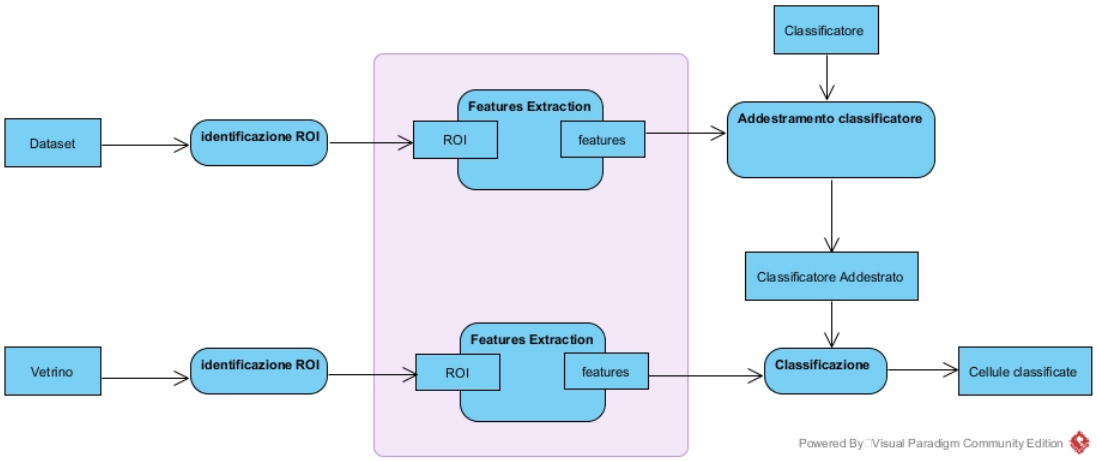

Figura 4.1 Architettura del funzionamento del sistema generale.

In Figura 4.1 è descritta l'architettura dell'intero sistema ideato. Seguendo lo sviluppo superiore del grafico, si può osservare come il sistema sia stato prima addestrato con vetrini di strisci di sangue che compongono il dataset e dai quali sono state ricavate migliaia di cellule, che sono state processate, per poter estrarre le features da fornire al classificatore per l'addestramento.

Nella parte inferiore invece, si sviluppa la parte operativa, del sistema. Avendo in input un vetrino scansionato al microscopio elettronico, si andranno ad identificare i ROI (Region Of Interest), che verranno processati per poterne estrarre il leucocita ed infine le features. Le features, verranno utilizzate per l'addestramento del classificatore, dopodiché sarà in grado di effettuare con una certa precisione la stima della classe di appartenenza del ROI individuato [18].

Per la progettazione del sistema è stato necessario suddividere logicamente il workflow in tre parti fondamentali: l'identificazione del ROI, nel vetrino, l'estrazione del leucocita e delle features, classificazione delle cellule, che comprende anche l'addestramento del classificatore. Inutile dire che ogni parte di questa suddivisione è propedeutica alla successiva, così come l'addestramento è propedeutico alla classificazione. Anche se i moduli sono funzionalmente dipendenti, sono stati sviluppati singolarmente, per mantenere un certo grado di modularità e favorirne l'ingegnerizzazione. I moduli nel loro insieme, non formano un sistema software completo e pronto per essere impiegato; bensì rappresentano l'ossatura sulla quale un sistema di questo tipo può essere creato.

Tale sistema è stato sviluppato e pensato, sia per essere impiegato con poco sforzo, sia come utilizzo ausiliario ad un altro sistema, in forma di libreria e perciò possibile utilizzare i moduli singolarmente, rispettando le dipendenze. Inoltre sono state effettuate alcune scelte tecniche ed implementative volte ad astrarre dalle tecnologie utilizzate, per facilitarne l'impiego; scelte che verranno in seguito argomentate, pur mantenendo un suo contesto di utilizzo.

L'obiettivo di questo lavoro, è quello di realizzare la seconda parte del sitema; relativa all'**estrazione del leucocita e delle features**.

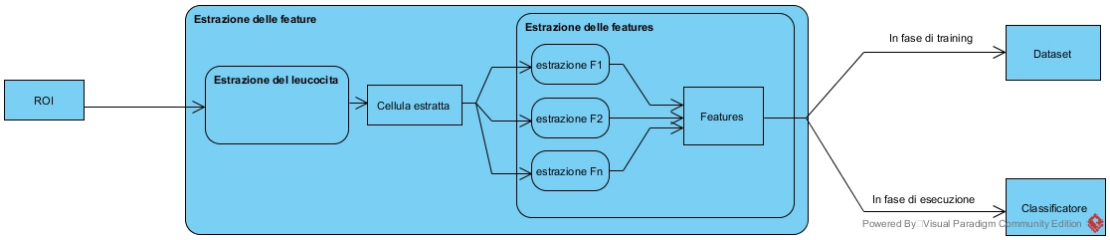

Figura 4.2 Architettura del lavoro di tesi.

L'estrazione delle feature, si può a sua volta dividere in identificazione ed estrazione del leucocita dal ROI ed estrazione vera e propria delle features, in cui ogni feature sarà indipendente dall'altra, le features successivamente, saranno poi utilizzate nella creazione del dataset in fase di training del classificatore, oppure andranno direttamente al classificatore durante l'esecuzione del sistema.

Di seguito si forniranno nozioni base sulle tecniche utilizzate e sulle strutture dati manipolate, per poi inoltrarci nello studio riguardante gli algoritmi utilizzati, la loro rielaborazione e le soluzioni proposte, che saranno debitamente motivate.

5 IMMAGINE DIGITALE

Un’immagine digitale è essenzialmente una rappresentazione bi-dimensionale tramite una serie di valori numerici.

Esistono due tipologie di immagini digitali:

- Bitmap: matrici di punti (o pixel)
- Vettoriali: insieme di punti (o nodi) uniti in linee o altre primitive grafiche (come cerchi, quadrati, triangoli)

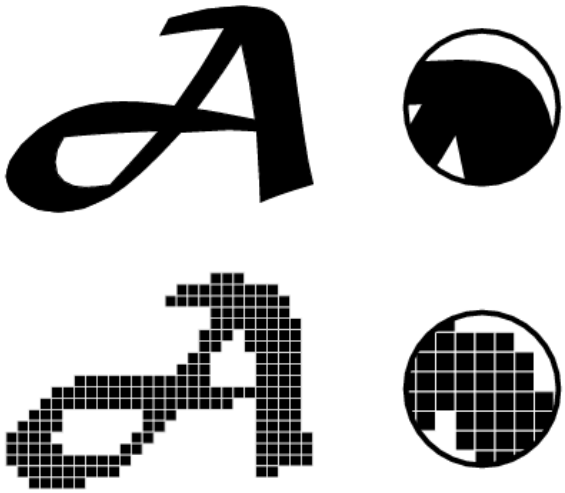

Figura 5.1 Un'immagine vettoriale (in alto), durante l'ingrandimento non si sgrana. Un'immagine bitmap (in basso) costituita da pixel, visibili durante l'ingrandimento.

In questo lavoro mi soffermerò solo sulla tipologia utilizzata cioè, immagini bitmap (o raster).

Le immagini bitmap si prestano a riprodurre le immagini della realtà, nelle loro sfumature di ombre e di colori. Si possono manipolare digitalmente e vengono generate da dispositivi di acquisizione come fotocamere digitali e scanner, ma anche con funzioni matematiche, attraverso l’utilizzo di software.

Utilizzano una griglia o retina di pixel per la rappresentazione e sono caratterizzate da:

- Dimensione in pixel: data dal numero di righe per il numero di colonne di pixel utilizzati
- Risoluzione: è il numero di pixel contenuti nell'unità di misura considerata (in genere pollice inglese, che misura 2.54 cm)
- Profondità di colore: il numero di bit dedicati ad ogni pixel per descrivere un singolo colore.

Quando le immagini vengono salvate digitalmente, vengono organizzate secondo schemi standard chiamati formati (come jpeg, png, tiff). In genere un formato è organizzato in due parti principali:

- Header: che contiene tutte le informazioni utili a leggere l'immagine, tra cui anche le caratteristiche descritte in precedenza.
- Dati: cioè la rappresentazione vera e propria dell'immagine composta da bit

Quando un'immagine viene salvata su disco, occupa uno spazio pari alla dimensione in pixel per profondità di colore. Se per esempio ho un'immagine di (modeste dimensioni) 1000x1000 pixel per 24 bit di profondità di colore (come nel caso dell'RGB, dove ogni canale possiede 8 bit di profondità), allora occuperò uno spazio in memoria di 2.86 Mbyte, cioè $24 * 10^{6}$ bit, che è uno spazio considerevole considerato che si tratta di una sola immagine, molto più piccola delle normali fotografie scattate da uno smartphone. Proprio per questo motivo i formati immagine, supportano algoritmi di compressione.

Un algoritmo di compressione, è un procedimento volto a ridurre lo spazio necessario a memorizzare un'immagine in memoria. La capacità di comprimere un'immagine, impiegata da un algoritmo di questo tipo, si misura con il cosiddetto **rapporto di compressione**. Il rapporto di compressione è essenzialmente il rapporto tra la dimensione di un file prima della compressione e la sua stessa dimensione a compressione avvenuta; maggiore è questo rapporto, più efficiente sarà il compressore, questa definizione è applicabile ad un qualsiasi compressore indifferentemente dal tipo di dati compressi. Esistono due tipologie di compressori, i **lossless**, che riducono lo spazio di memorizzazione, senza perdita della qualità originale, e i **lossy**, dove invece il file compresso ha una perdita di informazioni, ma ha in genere rapporti di compressione superiore al primo.

5.1 Formato PNG

Il formato PNG (Portable Network Graphics), è un formato di file per la memorizzazione di immagini. Nacque per l'uso sul web, ed è stato sviluppato in maniera simile al formato GIF, con lo scopo di soppiantarlo, in quanto i detentori del brevetto richiedevano il pagamento di una royalty per ogni programma che ne facesse uso.

Peculiarità del formato PNG sono: la possibilità di utilizzare trasparenza e opacità, l'uso di tavolozze di colore (supporta RGB a 24 bit e RGBA a 32 bit), immagini in scala di grigi, inoltre è ben indicato per immagini di dimensioni medio-piccole, in quanto a confronto con altri formati (come JPEG) non ci sono significative differenze di compressione (solo pochi Kb).

Il formato PNG utilizza un algoritmo di compressione lossless: DEFLETE; che è il risultato della combinazione dell'algoritmo LZ77, codifica di Huffman e di un filtro predittivo.

LZ77 si basa sulla sostituzione di parti dell'immagine che si ripetono più volte, sostituendo aree già incontrate con un puntatore e con la quantità di dati da copiare. L'algoritmo DEFLETE attua il filtro predittivo, sostituisce le parti duplicate (LZ77) ed infine effettua una sostituzione basata sulla frequenza d'uso dei simboli (codifica di Huffman).

In questo lavoro è stato utilizzato il formato PNG che è un formato di compressione lossless, al posto del più comune formato JPEG, (che è invece di tipo lossy), allo scopo di mantenere il livello informativo dei ROI (region of interest) che andrò ad analizzare. Infatti le immagini trattate hanno dimensioni dell'ordine di 50x50 / 100x100 pixel, quindi risulta importante evitare perdita di informazioni dovute alla memorizzazione di immagini per le fasi successive, inoltre come accennato in precedenza le ridotte dimensioni delle immagini su cui andrò a lavorare rendono poco significativi i benefici dovuti alla migliore compressione di altri formati.

5.2 Spazio dei colori

Per poter manipolare le immagini digitali, è necessario tener conto di come i colori reali siano rappresentati in forma numerica nei pixel. La gamma di colori possibili è molto ampia e può cambiare in base al dispositivo di acquisizione e in base alle necessità. Per rappresentare dei colori in informatica, si utilizza il cosiddetto **spazio dei colori**, cioè la combinazione di un modello di colore e di una funzione di mappatura idonea al modello.

Il modello di colore, è un modello matematico astratto, che descrive ogni singolo colore attraverso una combinazione di numeri, in genere di tre o quattro valori, detti *componenti colore*. Essendo un modello astratto, vengono introdotte specifiche regole, per adattarlo all'uso pratico.

La funzione di mappatura è una funzione matematica che fornisce una corrispondenza univoca tra un colore reale e una combinazione di componenti colore nello spazio dei colori [8].

In questo lavoro sono stati utilizzati come spazi di colore RGB e HSV, di seguito ne verranno descritte caratteristiche principali.

5.2.1 RGB

RGB è lo spazio di colori più utilizzato in computer graphics. Quasi ogni sensore, telecamera, fotocamera, scanner e schede grafiche, lo utilizzano. Si basa su tre livelli di colore da cui prende il nome rosso (Red), verde (Green), blu (Blue) ed è di tipo additivo, infatti sommando i tre livelli con intensità massima si ottiene il bianco, mentre combinandoli tra loro si ottengono il ciano, il magenta, il giallo ed il grigio se i tre livelli hanno la stessa intensità.

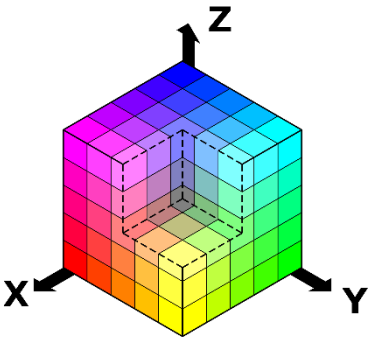

Figura 5.2 cubo RGB, sono visibili i 3 assi che costituiscono i colori principali.

Utilizza 8 bit su ogni livello, quindi dispone di 256 (2^8) combinazioni di colori per livello, per un totale di 16'777'216 (2^{8*3}) colori rappresentabili, che corrispondono a quasi tutto lo spettro visibile, nonostante l'occhio umano ne distingua non più di 10 milioni. Può essere rappresentato attraverso un sistema cartesiano a tre assi (in corrispondenza dei tre

colori principali), per ottenere un cubo unitario che ha ai vertici il bianco e il nero e nella diagonale principale i livelli di grigio.

5.2.2 HSV (HSI, HSL, HSB)

HSV (Hue Saturation Value), è un metodo additivo di composizione dei colori, basato sulla composizione di tre canali: tonalità saturazione e luminosità. HSV è simile a HSI, HSL, HSB, nei quali cambia la definizione di luminosità, rispettivamente definita come Intensity, Luminance e Brightness. Questo modello è particolarmente orientato sulla percezione umana di un colore in termini di tinta, e sfumature. Viene rappresentato su un sistema di coordinate cilindriche o coniche. Nei modelli cilindrici il Value è l'asse verticale del cilindro, Saturation è il raggio (va da 0 sull'asse a 1 sulla superficie), mentre Hue si misura da un angolo intorno all'asse verticale, con il rosso a 0 gradi, il verde a 120 e il blu a 240. Essendo la base del cilindro tutta dello stesso colore (nera), è possibile rappresentarlo come un cono rovesciato.

Lo spazio HSV è diffuso nell'image processing, e presenta notevoli vantaggi. In particolare il canale Hue è invariante rispetto alle ombre o alle lucentezze, e questa caratteristica è sfruttabile per superare effetti dovuti alla geometria e condizioni di illuminazione di un'oggetto rappresentato.

HSV presenta anche degli svantaggi; infatti Hue e Saturation non sono definiti per valori di Value prossimi o uguali a 0. Questa singolarità genera salti nei valori trasformati, creando instabilità numerica nei pressi della singolarità stessa e discontinuità nella rappresentazione dei colori.

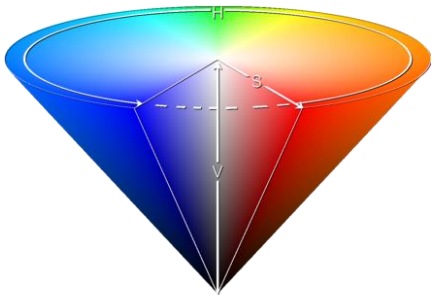

Figura 5.3 Cono HSV, il cono si ottiene a causa della singolarità presente per V tendente a 0.

6 DATASET

La creazione del dataset, è stato il primo fondamentale passo per l'inizio del progetto. I vetrini utilizzati, provengono dal U.O.C. di Ematologia dell'I.R.C.C.S. "Giovanni Paolo II" di Bari, in seguito sono stati analizzati e digitalizzati con un nuovo microscopio ottico prodotto dall'azienda Motic Europe, di Barcellona.

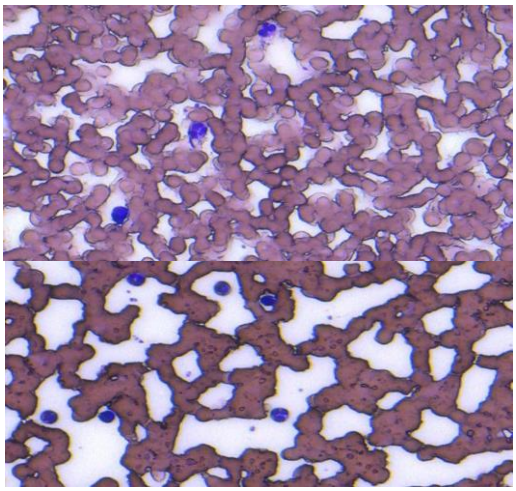

Figura 6.1 Regioni estratte da vetrini differenti, questi vetrini sono stati scartati a causa degli evidenti difetti di messa a fuoco.

Attraverso MoticDSAssistant software proprietario della Motic è stato possibile effettuare uno studio qualitativo sul vetrino, per stabilire quali vetrini fossero utilizzabili. È stata valutata la qualità in sé del vetrino come, la messa a fuoco, la tonalità cromatica e la luminosità, per stabilire la fattibilità e la bontà dei dati a nostra disposizione, considerando anche la possibilità di applicare filtri in grado di facilitare la manipolazione del vetrino.

Non è stato necessario applicare alcun filtro e con la valutazione di un specialista ematologo è stato deciso di utilizzare tre dei sei vetrini a disposizione, in quanto le cellule presenti erano più definite, c'era una chiara illuminazione di fondo, che permetteva la distinzione tra le cellule ed il nucleo, tuttavia era presente una leggera tendenza cromatica spostata verso il blu, dovuta al colorante utilizzato per evidenziare le cellule.

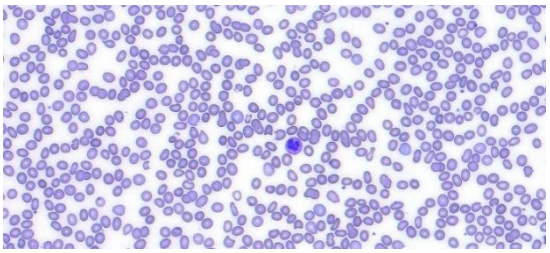

Figura 6.2 Vetrino utilizzato nel dataset, le cellule sono ben definite e non ci sono difetti di messa a fuoco, tuttavia la colorazione è tendente al blu.

Dai vetrini sono stati estratti all'incirca 18000 ROI (region of interest), che sono stati utilizzati come dataset per i test effettuati sull'estrazione della cellula e delle features.

6.1 Sistema di scansione

Il sistema di scansione utilizzato per la digitalizzazione del vetrino è il MoticEasyScanPro prodotto dall'azienda Motic, specializzata in strumenti di scansione ottica ad alta precisione. L'apparecchio è in grado di scansionare ed acquisire un vetrino con un ingrandimento massimo di 40x, di dimensioni 50000x50000 pixel. Vanta un sistema autofocus real time, con stabilizzazione del vetrino in quattro punti, elevata risoluzione a scansione progressiva meccanica e illuminazione a led incorporata. Il software prodotto dalla stessa azienda, DSAssistant Viever, permette di analizzare il vetrino con funzionalità avanzate di visualizzazione e misurazione. Permette di esportare il vetrino nei principali formati utilizzati per immagini di queste dimensioni tra cui il .ds, .jpg, jpg2000, e svs [6].

Nonostante l'avanzato sistema di scansione, alcuni vetrini presentano evidenti errori di messa a fuoco, anche se nel complesso le immagini si sono rivelate ottimo materiale di analisi.

Figura 6.3 Microscopio ottico MoticEasyScanPro.

7 TECNICHE DI SEGMENTAZIONE

La segmentazione di un'immagine digitale, è un processo di partizionamento in regioni il cui scopo è quello di rendere l'immagine più significativa sotto un aspetto arbitrario e sotto tale aspetto più facilmente analizzabile. Il risultato di un'immagine segmentata è un insieme di regioni (chiamate segmenti), comprendenti dei pixel che vengono etichettati tutti nello stesso modo e perciò assegnati a quella porzione d'immagine. Come conseguenza di tale processo di "etichettatura", tutti i pixel etichettati allo stesso modo, possiedono proprietà simili tra loro secondo la caratteristica scelta, come per esempio colore, intensità o tessitura (cioè la distribuzione spaziale percepita guardando una regione di pixel). Questo procedimento ha molte applicazioni nel medical imaging e può essere utilizzato anche per la ricostruzione tridimensionale di organi o parti del corpo.

Al termine del processo di etichettatura (labeling) generalmente si ottiene un'immagine binaria, in cui i pixel hanno valori 0 o 1, di solito quest'ultimo è utilizzato come marcatore dell'oggetto che si sta evidenziando. Successivamente si può sovrapporre quest'immagine all'immagine originale, come una sorta di maschera, per poter selezionare le regioni significative evidenziate dalla maschera stessa.

In letteratura esistono svariati metodi di segmentazione, secondo [2] possono essere classificati in:

- Metodi basati su edge detection
- Metodi basati sull'analisi dell'istogramma (sogliatura)
- Metodi basati su grafi e/o alberi
- Metodi basati su region splitting e region growing
- Segmentazione basata sul modello
- Segmentazione basata su reti neurali
- Metodi di partizionamento dei grafi
- Trasformata Watershed
- Segmentazione multi scala
- Metodi basati su approccio probabilistico e bayesiano

In questo lavoro sono stati utilizzati metodi basati sull'analisi dell'istogramma, metodi basati su grafi e partizionamento dei grafi. Di seguito saranno descritte le principali caratteristiche degli algoritmi utilizzati.

7.1 Thresholding

Il thresholding o sogliatura, si basa come già detto sull'analisi dell'istogramma del colore. L'istogramma del colore, si ottiene ponendo sull'asse delle ascisse tutti i colori presenti nell'immagine e sull'asse delle ordinate il numero di occorrenze di ogni singolo colore. Dall'istogramma si calcola una soglia chiamata *threshold*. La suddivisione avviene attribuendo un'etichetta a tutti i valori minori della soglia e un'altra etichetta a tutti i restanti valori. In questo modo si ha come risultato un'immagine binaria, suddivisa in due partizioni. Naturalmente il tutto consiste nella scelta della soglia, che deve essere automatica ed in questo si distinguono gli algoritmi di thresholding.

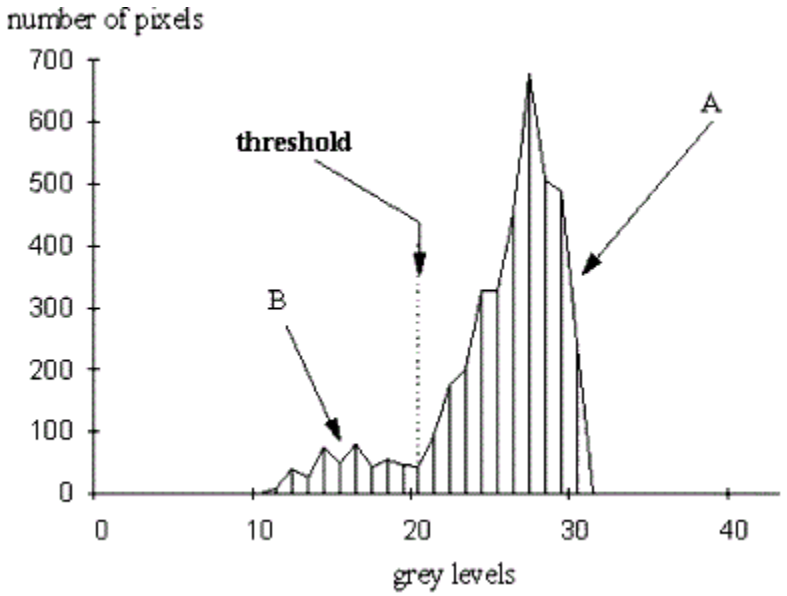

Figura 7.1 Istogramma del colore, il treshold divide i pixel in due regioni.

7.1.1 Metodo di Otsu

Esistono diversi algorimti di thresholding; in questo lavoro ho utilizzato il **metodo Otsu**.

Il metodo Otsu presume che l'immagine contenga due sole classi e calcola la soglia ottimale per separarle. La suddivisione ottimale si ottiene massimizzando la varianza inter classe, ma Nabojuki Otsu dimostrò che questo è equivalente a minimizzare la varianza intra classe [7].

La minimizzazione della varianza intra classe, può essere definita come la somma pesata delle varianze delle due classi e i pesi rappresentano la probabilità che le due classi siano separate dalla soglia *t* e dalla varianza. Invece nella massimizzazione, i pesi dipendono solo dalla media. Calcolare la media è molto più semplice che calcolare la varianza, perciò su tutti valori dell'istogramma, si andranno a testare delle soglie in un certo intervallo di valori e si aggiorneranno i pesi e la media, che potranno essere utilizzati per calcolare la varianza intra classe. Al termine dell'operazione, si manterrà solo la soglia che ha generato la varianza più alta, tale soglia sarà anche quella che minimizza la varianza intra classe.

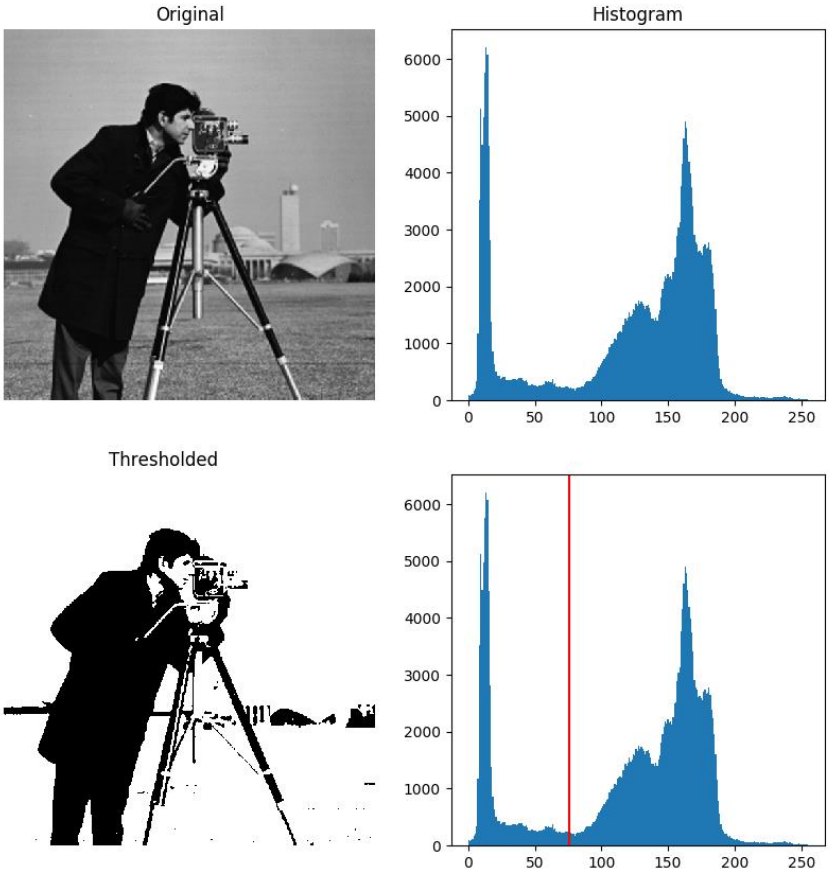

Figura 7.2 Applicazione del metodo di Otsu.

Il metodo Otsu tuttavia ha delle limitazioni: non funziona su immagini monocromatiche, o su immagini con più di un canale (come RGB). Infatti può essere applicato a immagini in scala di grigi o ai singoli canali di un'immagine RGB.

8 OPERAZIONI MORFOLOGICHE

Le immagini binarie di solito, contengono numerose imperfezioni; l'image processing morfologico, si pone l'obiettivo di rimuovere queste imperfezioni tenendo conto della forma e della struttura dell'immagine. La base teorica di queste tecniche proviene dalla morfologia matematica (mathematical morphology), che contiene tutte le tecniche per poter manipolare strutture geometriche basate su teoria degli insiemi, teoria del reticolo, topologia e random functions. Le operazioni morfologiche, lavorano su immagini binarie (composte da 0 e 1) e si sviluppano attraverso le più semplici operazioni booleane, ma possono essere applicate anche a immagini in scala di grigi.

L'immagine può essere vista come una matrice nel quale possiamo trovare pixel di valore 0 o 1, questa matrice è di fatto uno *spazio euclideo*. Inseriamo tutti i pixel di valore 1 in un insieme A, e tutti gli altri pixel di valore 0 si troveranno nell'insieme A^c. L'idea di base è quella di utilizzare una forma predefinita chiamata *elemento strutturale* come sonda, per effettuare delle operazioni tra A e B. L'*elemento strutturale* è esso stesso un'immagine binaria ed è possibile definirlo a piacimento.

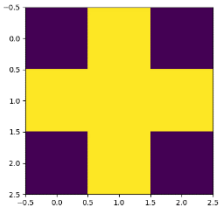

Figura 8.1 elemento strutturale.

Le operazioni morfologiche di base sono: erosione, dilatazione, apertura e chiusura. Tutte queste operazioni sono state utilizzate negli algoritmi presentati, perciò andrò a descriverne le caratteristiche principali.

8.1 Erosione

L'erosione detta anche sottrazione di *Minkowski* è definita come:

$$A \ominus B = \left\{ c \in E^2 : c + b \in A, \quad \forall b \in B \right\}$$

Cioè A erosione B è formato da tutti i punti tali che aggiungendo all'oggetto eroso qualunque punto dell'elemento strutturale ottengo punti che appartengono ad A.

In altre parole, B viene passato come una sonda su A, centrandolo con il bordo di A (la fascia di punti più esterna). A questo punto viene sottratto B da A (cioè vengono eliminati da A i punti in comune con B). Questa operazione viene effettuata su tutto il bordo di A.

In genere si utilizza un elemento strutturale tipicamente simmetrico, come un disco o un quadrato. L'effetto dell'erosione è quello di contrarre e rimpicciolire gli oggetti dell'immagine in maniera isotropica (indipendentemente dalla direzione). L'erosione può essere utilizzata, per dividere regioni debolmente connesse e per definire o distinguere regioni tra loro.

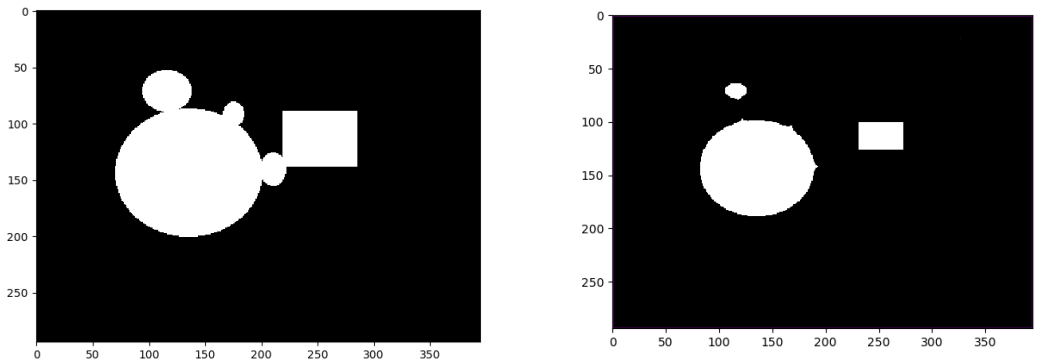

Figura 8.2 L'immagine originale (a sinistra), mentre l'immagine erosa (a destra) con disco di 15 pixel.

8.2 Dilatazione

La dilatazione detta anche somma di *Minkowski* è definita come:

$$A \oplus B = \left\{ c \in E^2 : c = a + b, \quad a \in A, \quad b \in B \right\}$$

Cioè A dilatazione B è l'insieme di tutti i possibili punti che otterrei sommando B a tutti i punti di A.

In altre parole in maniera analoga alla precedente, B viene passato come una sonda su A, centrandolo con il bordo esterno di A (la fascia vuota di punti più esterna). Solo che in questo caso B viene aggiunto ad A (cioè vengono aggiunti ad A i punti non in comune). Questa operazione viene effettuata su tutto il bordo di A.

Anche in questo caso l'elemento strutturale generalmente è simmetrico. L'effetto della dilatazione è un'espansione isotropica dell'oggetto. Questa operazione può essere utilizzata per rafforzare regioni debolmente connesse o coprire alcune lacune generate dalla segmentazione. Si può usare per esempio per unire regioni che rappresentano frammenti di uno stesso oggetto.

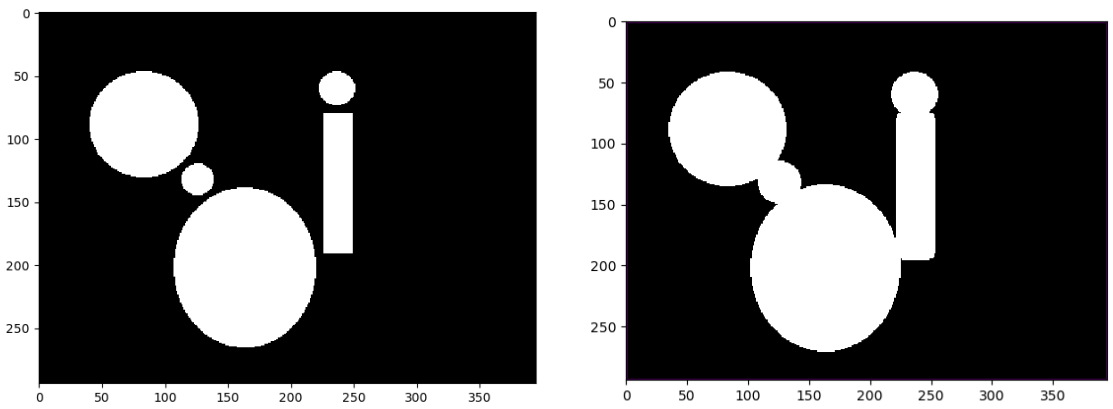

Figura 8.3 L'immagine originale (a sinistra), l'immagine dilatata (a destra) con un disco di 10 pixel

8.3 Apertura

L’apertura (opening) è la successione di una erosione e di una dilatazione con lo stesso elemento strutturale:

$$A \circ B = (A \ominus B) \oplus B$$

È utile per effettuare una pulizia del rumore (*denoise*), come piccole aree o punti superflui ottenuti dalla segmentazione. Scegliendo un elemento strutturale appropriato, assolve alla funzione di filtro, eliminando zone non rilevanti

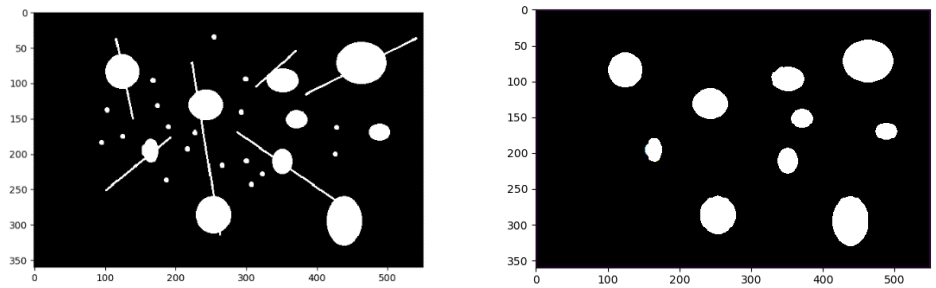

Figura 8.4 L'immagine originale (a sinistra), l'apertura effettuata (a destra) con un disco di 5 pixel.

8.4 Chiusura

La chiusura (closing) è la successione di una dilatazione e di un’erosione con lo stesso elemento strutturale:

$$A \bullet B = (A \oplus B) \ominus B$$

L’effetto della chiusura è quello compattare regioni fortemente connesse e quindi rafforzarle. Per esempio è utile per riempire piccoli buchi e concavità nelle regioni.

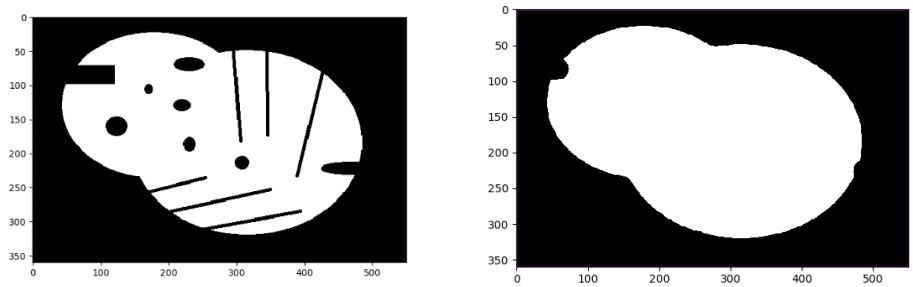

Figura 8.5 L'immagine originale (a sinistra), la chiusura effettuata (a destra) con un disco di 15 pixel.

8.5 Trasformata Hit-and-Miss

La trasformata hit-and-miss è l'operazione morfologica alla base di tutte le operazioni morfologiche, in quanto esprime a pieno la capacità di "sondare" dell'elemento strutturale. Come altre operazioni morfologiche prende in input un'immagine binaria e un elemento strutturale, restituendo un'immagine binaria.

Nell'hit-and-miss così come in tutte le altre operazioni morfologiche, l'elemento strutturale è una forma primitiva. Essa viene rappresenta in forma matriciale per operare sull'immagine (che è anch'essa una matrice) e prende il nome di **kernel**. Il concetto di kernel, è molto utilizzato nell'image processing anche per altre operazioni come il filtering che verrà dettagliato più avanti.

L'hit-and-miss, confronta il proprio kernel o elemento strutturale, su ogni pixel dell'immagine, quando ottiene una corrispondenza esatta (*matching*) tra il kernel e l'immagine, imposta il pixel al centro del kernel con un marcatore. Se il *matching* non ha riscontro positivo tutti i pixel sottostanti vengono settati con il colore dello sfondo.

Se per esempio, costruisco degli elementi strutturali per identificare un angolo in un oggetto, l'algoritmo hit-and-miss, mi restituirà un'immagine booleana in cui saranno evidenziati con il marcatore solo i punti in cui ha individuato gli angoli.

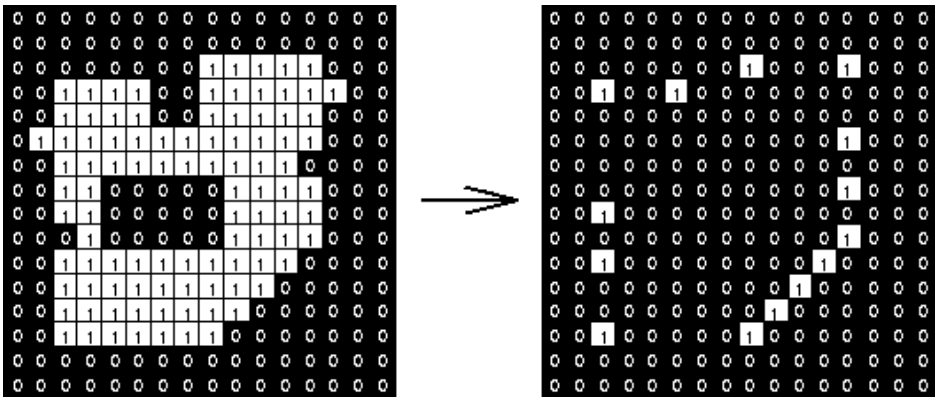

Figura 8.6 L'immagine originale (a sinistra), il risultato della trasformata hit-and-miss (a destra) con elemento strutturale per la rilevazione degli angoli.

Questo operatore è utile per individuare pattern, bordi, o particolari strutture all'interno degli oggetti di un'immagine booleana. Viene utilizzato in algoritmi più avanzati come lo **skeletonize** e il **thinning** [11].

8.6 Skeletonize

L'algoritmo di scheletrizzazione (**skeletonize**) è un processo di riduzione degli oggetti di un'immagine binaria ad uno scheletro che in larga misura conserva la connettività e l'entità della regione eliminando la maggior parte dei pixel. Il processo è assimilabile ad un'oggetto che lentamente "brucia" da tutti i fronti simultaneamente, consumandolo fino ad estinguersi quando i fronti di fuoco si incontrano [12]. Il risultato è quindi un insieme di segmenti, che costituiscono lo scheletro dell'immagine, lo scheletro risultante chiamato anche *scheletro topologico* enfatizza le proprietà topologiche e geometriche della forma originaria, come connettività, topologia, lunghezza, direzione e larghezza mantenendo tutte le informazioni necessarie per ricostruire l'oggetto originario.

In realtà l'algoritmo di **skeletonize**, [17] deriva dal così detto **thinning**. Il thinning è un algoritmo legato alla trasformata **hit-and-miss**, tanto da poter essere totalmente espresso attraverso quest'ultima.

Il thinning di un'immagine A da un elemento strutturale B definito come:

$$thin(A, B) = A - hitAndMiss(A, B)$$

Dove la sottrazione è la *sottrazione logica* definita come: $X - Y = X \cap \neg Y$

Il thinning ha come effetto quello di assottigliare l'oggetto, consumandolo dai bordi. Questa operazione in genere viene eseguita un numero limitato di volte. Lo skeletonize, esegue questa operazione fino a quando l'oggetto non diventa totalmente lineare, cioè costituito da soli segmenti, quindi la principale differenza risiede nel numero di iterazioni eseguite

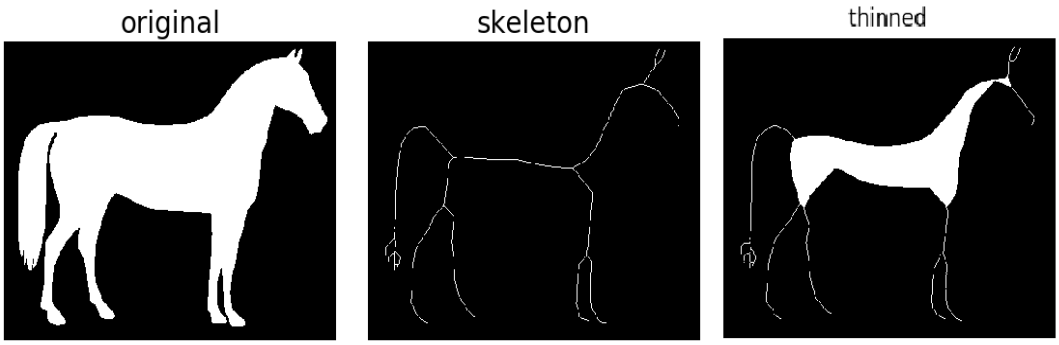

Figura 8.7 risultato dello skeletonize e del thinning. Il thinning può comunque esser applicato fino ad ottenere lo stesso risultato dello skeletonize.

8.7 Trasformata Medial Axis

Anche la **trasformata medial axis** (MAT), ottiene un risultato simile a allo skeletonize, ma con una tecnica differente. La prima differenza sta nel fatto che restituisce un'immagine in scala di grigi e non un'immagine binaria, questo perché oltre allo scheletro, calcola la distanza di ogni pixel dello scheletro dal bordo dell'oggetto iniziale, e lo rappresenta con un gradiente di colore.

È possibile calcolarla in molte maniere; descriverò il metodo basato sulla **trasformata distanza** [1] in quanto è lo stesso metodo implementato nella libreria che ho utilizzato per l'implementazione degli algoritmi. La trasformata distanza (distance transform) è un operatore che normalmente non viene applicato su immagini binarie. Effettua una trasformazione sull'immagine convertendola in un *campo di distanze*, cioè viene calcolata la distanza di ogni pixel dell'oggetto dal bordo dell'oggetto stesso, quindi ogni pixel viene avvalorato con la sua distanza dal bordo, ottenendo in output un'immagine con la stessa forma, ma in scala di grigi.

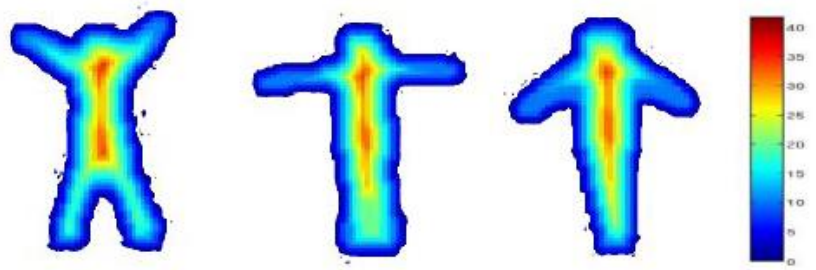

Figura 8.8 trasformata distanza applicata a tre immagini differenti. I colori tendenti al rosso sono gli elementi più distanti dai bordi.

Applicando la trasformata distanza all'immagine, risulta evidente come lo scheletro, risieda negli elementi più distanti dal bordo, cioè quelli che hanno valori più alti, che visivamente sono le pieghe e le curvature dell'immagine risultante.

9 FILTRAGGIO

Il filtraggio è una tecnica fondamentale per l'image processing, e può essere utilizzata per il miglioramento dell'immagine o per ottenere effetti particolari che ne risaltino determinate caratteristiche. Il concetto di filtro nasce in campo elettronico nella teoria dei segnali, dove il filtro è considerato come un operatore o funzione che prende in input un segnale e ne restituisce un altro. Esistono svariate tipologie di filtri che si possono classificare in base a particolari aspetti, in questo lavoro è stato utilizzato un filtro lineare.

Un filtro lineare, ha come vincolo quello della linearità, cioè dato un segnale in ingresso, il segnale in uscita sarà combinazione lineare del primo. Tale concetto è molto generale e si applica in campo elettronico, meccanico, statistico e informatico. Nell'elaborazione di immagini digitali [9], un filtro si presenta come una funzione che viene applicata ad ogni pixel dell'immagine, tale funzione prende il nome di *funzione di intorno*, in quanto agisce tenendo conto anche dei pixel circostanti a quello interessato (*finestra di punti*), perciò il filtro è detto *operatore locale.* Per i filtri lineari, la finestra di punti appena menzionata si definisce come **nucleo di convoluzione** (convolution kernel), è si presenta come una matrice di dimensioni $I \times J$ con all'interno dei pesi che determinano la funzione di filtraggio.

In questo lavoro è stato utilizzato solo il filtro gaussiano, e di seguito ne descriverò le caratteristiche principali.

9.1 Filtro Gaussiano

Il filtro Gaussiano, prende il nome dalla distribuzione gaussiana, che linearmente viene descritta come:

$$G(x) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}$$

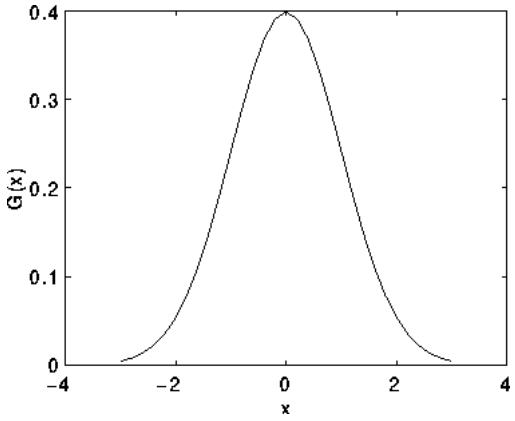

Figura 9.1 Distribuzione Gaussiana.

Dove σ è la deviazione standard della distribuzione. Questa distribuzione può essere espressa come una isotropica in una superficie:

$$G(x,y) = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

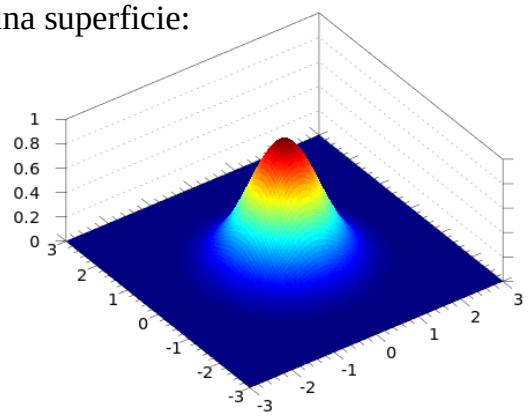

Figura 9.2 Distribuzione Gaussiana in 3d.

Per poter produrre il filtro [10] abbiamo la necessità di riprodurre la gaussiana su un nucleo di convoluzione (kernel). Il problema principale risiede nel fatto che la distribuzione gaussiana non è mai nulla, quindi il nostro kernel dovrebbe avere dimensione infinita. Ma in pratica la distribuzione tende a zero superati 3 σ (deviazioni standard) dalla media; quindi è possibile troncare il kernel in questo punto. Il kernel tuttavia è costituito da valori interi, ed è necessario discretizzare la gaussiana. La scelta dei valori non è un problema banale, uno dei kernel più utilizzati è il *Kernel Gaussiano Discreto* in cui viene inserito al centro del kernel il pixel oggetto del filtro e viene integrata la gaussiana (sommando il suo valore a incrementi di 0.001). I valori ottenuti non sono interi, ma si possono normalizzare, in modo da ottenere il valore 1 agli angoli del

kernel, inoltre la normalizzazione ha l'effetto di rendere il filtro indipendente dalla luminanza dell'immagine.

$$\frac{1}{273}\begin{bmatrix} 1 & 4 & 7 & 4 & 1 \\ 4 & 16 & 26 & 16 & 4 \\ 7 & 26 & 41 & 26 & 7 \\ 4 & 16 & 26 & 16 & 4 \\ 1 & 4 & 7 & 4 & 1 \end{bmatrix}$$

Figura 9.3 Kernel o nucleo di convoluzione del filtro Gaussiano.

Il risultato di questo filtro è uno *smoothing* dell'immagine, una sorta di sfocatura, che aumenta all'aumentare del σ scelto in quanto la dimensione del kernel cresce e come conseguenza se ne amplificano gli effetti.

Il filtro gaussiano permette di ridurre il rumore dell'immagine, questo si traduce in un istogramma del colore smussato e con pochi sbalzi. Quindi risulta fondamentale il suo utilizzo nell'analisi dell'istogramma per le operazioni di thresholding, in quanto riduce notevolmente l'errore dovuto al rumore.

Figura 9.4 Applicazione del filtro gaussiano per differenti σ.

10 LABELING

Il labeling è un algoritmo che effettua un processo di etichettatura di tutte le componenti connesse di un'immagine binaria, utilizzando un marcatore. Al termine del processo, tutte le regioni connesse verranno considerate come un'unica regione, mentre le regioni non connesse, saranno distinte tra loro grazie a marcatori differenti.

In un'immagine una regione si dice *connessa*, se presi due punti qualsiasi, esiste un percorso fra loro interamente contenuto nella componente stessa. Il concetto di percorso non è banale, infatti il percorso si costruisce verificando se due pixel o punti della regione sono adiacenti. Per verificare questa condizione, bisogna definire una **metrica.**

Una **metrica**, è una funzione che indica come viene calcolata la distanza tra due punti, è permette di determinare se i due punti sono adiacenti, rispetto a tale regola.

Esistono due metriche: $\mathbf{D_4}$ che permette solo spostamenti orizzontali e verticali e la $\mathbf{D_8}$ che ammette anche spostamenti in diagonale.

Quindi a seconda della metrica scelta si parla di regioni con **4-connessione** o **8-connessione** [8].

Possiamo definire una **componente connessa** o **blob** di un'immagine binaria la massima regione di punti marcati come oggetto, che soddisfano la metrica utilizzata.

L'algoritmo di labeling, si articola in 2 step [2]:

1. Nel primo step viene scansionata l'immagine, alla ricerca di pixel connessi e vengono attribuite delle etichette temporanee. Infatti in uno stesso oggetto ci potrebbero essere pixel con etichette diverse. Ciò è causato dal fatto che la scansione è sequenziale e potrebbe essere effettuata per righe o per colonne. Viene creata una tabella che contiene le classi di equivalenza, che per ora conterranno solo i pixel della prima scansione con le rispettive etichette.
2. Nella seconda scansione (che deve essere effettuata in maniera diversa dalla prima), gli elementi scansionati, vengono etichettati allo stesso modo, in seguito sono confrontati con quelli presenti nella tabella. Da questo controllo incrociato, è possibile rimuovere le ridondanze e correggere le etichette errate, in quanto si avrà una visione completa del neighborhood di ogni pixel.

11 PATTERN RECOGNITION

Il pattern recognition è una branca del machine learning (apprendimento automatico) e si occupa del riconoscimento di modelli (patterns) e regolarità all'interno di dati. Il pattern recognition è anche legato all'intelligenza artificiale ed è molto utilizzato in computer vision. Molti sono i problemi riconducibili al pattern recognition, ma in un senso molto più generale, il pattern recognition, si occupa di fornire risposte ragionevoli, per tutti i possibili input, dando un riscontro "molto probabile" e tenendo conto della varianza statistica dei dati. Questo processo si differenza dal pattern matching, che invece si propone di trovare un esatto riscontro all'interno dei dati in input, considerando pattern già esistenti.

11.1 Fitting

Uno dei problemi di base del pattern recognition e del computer vision, è il *fitting*: cioè l'adattamento di primitive geometriche in un insieme di punti. Il problema si riduce alla ricerca di una funzione matematica, che abbia la migliore corrispondenza ad un insieme di punti assegnati (possibilmente limitati). Tali punti verranno perciò distinti in **inliers** e **outliers**; i primi sono quelli appartenenti al modello, i secondi sono punti estranei considerati di disturbo (*noise*).

L'utilizzo di primitive geometriche (come curve, cerchi o ellissi), permette la riduzione e la semplificazione di dati, che risultano in seguito più facili da trattare. In letteratura esistono metodi per il rilevamento ed il fitting, basati su tecniche differenti. Secondo [4], si possono dividere in due tipologie generali: quelli basati su metodi iterativi di stima di parametri e quelli basati sull'ottimizzazione di una funzione obiettivo; i primi risultano robusti agli outliers, ma sono lenti e richiedono molta memoria e la loro accuratezza può essere bassa, i secondi hanno maggiore velocità ed accuratezza, ma hanno lo svantaggio di essere sensibili agli eventuali outliers (poco robusti).

L'immagine di un globulo bianco, può avere colore, forma e dimensione differenti, perciò trattasi di input con alto numero di outliers. Nei test si sono adottati algoritmi relativi alla prima categoria delle due appena descritte, considerando il fatto che la maggiore accuratezza prodotta dalla seconda categoria sarebbe stata annullata dalla massiccia presenza di outliers.

11.2 RANSAC

RANSAC (random sample consensus) è un metodo iterativo per la stima di parametri di un modello a partire da un insieme di dati ed è largamente utilizzato in computer vision. È un metodo non deterministico, cioè produce un risultato corretto con una certa probabilità, che aumenta al numero di iterazioni eseguite. Si basa sull'assunzione che i dati si possano dividere in inliers e outliers e che dato un insieme di inliers, esiste una procedura per poter stimare dei parametri di un modello che può interpretare i dati. Di seguito verrà spiegato brevemente il suo funzionamento.

RANSAC usa un sistema di votazione per trovare il risultato ottimale. Ogni elemento del dataset, viene utilizzato per dare un voto ad uno o modelli multipli. L'implementazione del meccanismo di voto si basa su due concetti fondamentali: i dati che rappresentano rumore (outliers) non voteranno in modo consistente per nessun modello, mentre ci sarà una buona quantità di dati, che invece concorderà con il modello (inliers). L'algoritmo è composto essenzialmente da due step ripetuti iterativamente [19]:

1) Un insieme di dati selezionati casualmente dal dataset, vengono utilizzati per stimare i parametri di un modello (*ipotetici inliers*). La cardinalità dell'insieme di dati scelto dovrà essere sufficiente a stimare il modello.
2) L'algoritmo verifica quali elementi dell'intero dataset sono consistenti con il modello creato nel primo step. Un elemento del dataset è considerato outlier se non si adatta al modello e viene stimata una soglia di errore che definisce la massima deviazione che un elemento deve avere per essere considerato rumore. La stima dell'errore che identifica gli outliers, viene effettuata attraverso una funzione costo.

L'insieme di inliers ottenuto è chiamato *insieme di consensi*. RANSAC esegue questi due passi iterativamente fino a quando l'insieme dei consensi non contiene abbastanza inliers. Al verificarsi di questa condizione, i parametri del modello vengono stimati nuovamente con l'intero *insieme dei consensi*.

Questo processo viene eseguito un numero di volte prefissato, per produrre diversi modelli. A questo punto i modelli migliori saranno quelli che avranno più consensi e quindi anche un maggior numero di elementi appartenenti al proprio *insieme dei consensi*.

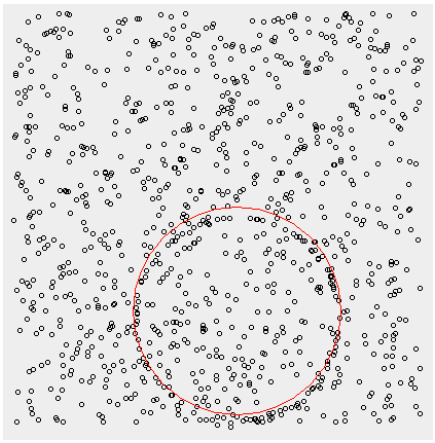

Figura 11.1 Applicazione di un modello circolare ad un insieme di punti utilizzando RANSAC.

11.3 Trasformata Di Hough

La trasformata di Hough permette di identificare forme primitive descritte da equazioni analitiche, trasformando il problema della ricerca di una curva, in un problema di massimizzazione. In generale una curva piana è definita in forma analitica tramite un insieme limitato di parametri. Quindi la funzione della curva lega le coordinate cartesiane ai parametri della curva. Allo stesso modo un punto nello spazio dei parametri individua univocamente una curva analitica.

In altre parole ogni punto dello *spazio immagine* corrisponde ad una ipersuperficie (superficie generalizzata) nello *spazio dei parametri*. Ciò significa che *n* punti nello *spazio immagine* appartenenti ad una stessa curva, generano *n* superfici che si intersecano in uno stesso punto nello *spazio dei parametri*.

Esistono due proprietà duali da questa affermazione:

- Ogni punto nello *spazio dei parametri* corrisponde ad una singola istanza della curva nello *spazio immagine*
- *n* punti dello nello *spazio dei parametri* appartenenti ad una stessa ipersuperficie, corrispondono a *n* curve che si intersecano in uno stesso punto nello *spazio immagine*

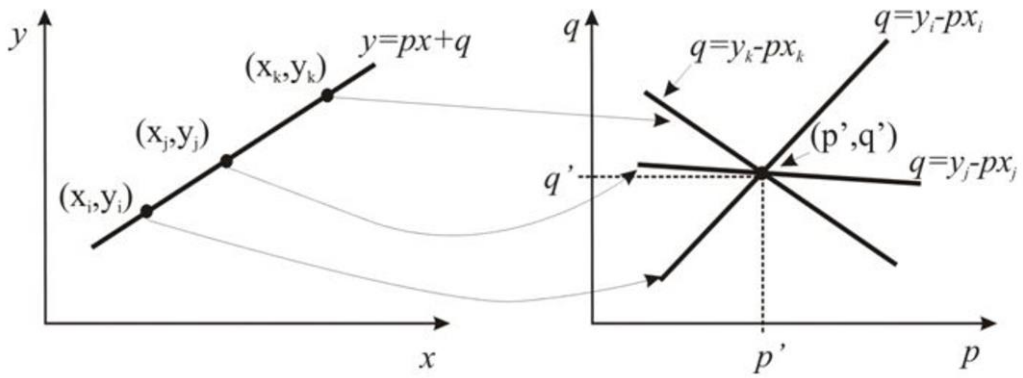

Figura 11.2 Lo spazio immagine (a sinistra), lo spazio dei parametri (a destra). 3 punti di una stessa retta nello spazio immagine corrispondono a 3 rette nello spazio dei parametri e il punto di intersezione corrisponde alla retta nello spazio immagine.

Se ogni punto nello *spazio immagine* genera una superficie nello *spazio dei parametri* allora nello *spazio dei parametri* una intersezione di molte superfici è l'indizio della presenza di una particolare istanza della curva analitica cercata.

Occorrerà un numero di punti almeno pari al numero dei parametri della curva per caratterizzarla.

In sostanza la trasformata di Hough, permette di convertire un problema di ricerca di curve in quello più semplice di ricerca di intersezioni. Il problema di massimizzazione consiste nel individuare nello *spazio dei parametri* il punto con il maggior numero di intersezioni. Tale punto ha come coordinate una tupla di parametri, che caratterizzano la curva cercata.

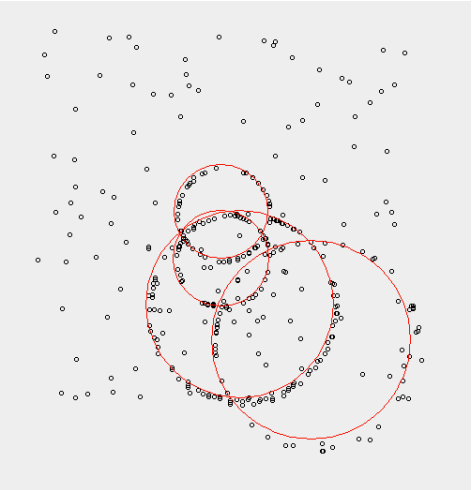

Figura 11.3 I migliori 4 modelli individuati utilizzando la trasformata di Hough.

Ogni curva che si interseca in un punto può essere considerata come un voto in favore di quel punto. I voti possono essere pesati in base alla loro significatività e generalmente viene fissata una soglia al numero di voti, che determina la curva migliore. Questa soglia si può determinare attraverso criteri teorici (come rapporto segnale / rumore) o come compromesso fra il rischio di non rilevare oggetti o rilevarne troppi (falsi positivi). Nell'implementazione della trasformata vengono utilizzati operatori differenziali (come Sobel), che producono informazioni sulla direzione del gradiente, allo scopo di ottimizzare la ricerca.

11.4 Canny Edge Detector

L'algoritmo di Canny è considerato il miglior algoritmo per il rilevamento dei contorni (**edge detection**) in un'immagine in scala di grigi, in quanto riesce a individuare il maggior numero di contorni con molta precisione, pur rimanendo poco sensibile al rumore.

L'algoritmo di Canny per diminuire la sensibilità al rumore utilizza un filtro basato sulla derivata prima del filtro gaussiano. A questo punto il modo migliore per individuare i contorni è considerare la direzione della luce all'interno dell'immagine. Vengono usati quattro filtri differenti per ogni direzione: orizzontale, verticale, e diagonale dell'immagine, per ogni pixel risultante, viene considerata corretta la direzione corrispondete al filtro che dà il valore maggiore. Il processo di individuazione della direzione luminosa, corrisponde alla ricerca del massimo gradiente della luminanza di ciascun punto dell'immagine e viene ottenuta la cosiddetta *mappa dei gradienti*

dell'immagine. I tratti individuati nella mappa dei gradienti, corrispondono a punti di forte intensità dell'immagine originale, ciò significa che vi è una forte probabilità della presenza di un contorno. Per effettuare una stima più precisa, si attua la cosiddetta *soppressione dei non-massimi*, cioè per ogni tratto contiguo, si va alla ricerca dei massimi locali (dove il gradiente si annulla) e si considerano solo tali punti nella fase successiva.

Nell'ultimo step si effettua una fase di sogliatura chiamata *sogliatura con isteresi*: si definiscono due soglie e si confrontano con i punti del gradiente; se il gradiente è inferiore alla soglia più bassa il punto viene scartato, se è superiore a quella alta, viene accettato, se si trova tra le due soglie, viene accettato solo se *contiguo* ad un altro punto precedentemente accettato. Al termine dello step, si ottiene una mappa binaria, contenente tutte le linee dei contorni riconosciuti. È possibile personalizzare l'algoritmo modificando parametri come la dimensione del filtro gaussiano e le soglie applicate.

Figura 11.4 Immagine di una Alfa Romeo Disco Volante (a sinistra), i contorni rilevati (a destra) dopo l'applicazione del Canny Edge Detector.

12 ESTRAZIONE DEL LEUCOCITA

La prima fase del lavoro di tesi, ha lo scopo di estrarre l'immagine del leucocita dal ROI, ottenuto dalla fase precedente. Il ROI (region of interest) è un'immagine di una porzione di vetrino, dove si presume si trovi un leucocita

Figura 12.1 ROI contenente un leucocita

Per poter estrarre la cellula, è quindi necessario isolarla completamente eliminando il background e le altre cellule o parti di cellule che non appartengono al leucocita. Come primo approccio, per la realizzazione di questa fase, ho implementato l'algoritmo proposto da [8], del quale descriverò brevemente lo schema logico:

1. Conversione del ROI da RGB a scala di grigi
2. Rimozione dello sfondo dalla finestra tramite sogliatura T sull'istogramma del colore
3. Individuazione dei contorni delle cellule con metodo del triangolo, sul canale blu
4. Rimozione del rumore presente nei contorni
5. Chiusura con disco di dimensione 12 pixel
6. Unione dell'immagine di sfondo con i contorni
7. Eliminazione delle regioni esterne alla zona centrale del ROI
8. Aggiunta dei pixel del nucleo
9. Rimozione buchi del globulo bianco
10. Apertura morfologica con disco di dimensione 11 pixel
11. Applicazione filtro mediano

Considerato che l'algoritmo è stato adattato alle immagini che avevo a disposizione, non ha prodotto buoni risultati se non per pochissime cellule, in quanto lo step 3 e 5 risultavano poco robusti e producevano molti errori nell'individuazione dei contorni delle cellule circostanti.

Il problema fondamentale risiede nell'irregolarità dell'istogramma del colore, che risulta molto variabile a seconda della cellula, senza considerare le variazioni di colore presenti su immagini estratte dallo stesso vetrino o la possibilità di cellule adiacenti.

12.1 Ipotesi

Studiando a fondo le motivazioni del fallimento dell'algoritmo precedente e gli algoritmi presenti in letteratura, ho formulato quattro ipotesi sfruttando concetti e approcci molto semplici, ma diversi tra loro. Perciò ho deciso di suddividerle logicamente in due idee fondamentali:

1. Individuare direttamente il leucocita, effettuando operazioni atte a evidenziale esplicitamente i pixel relativi ad esso.
2. Individuare indirettamente il leucocita andando alla ricerca di pixel complementari, cioè non appartenenti al leucocita, come altre cellule, rumore o oggetti estranei, per poi rimuoverle.

La prima idea è l'approccio più logico in questo genere di problemi ed è anche stato utilizzato nell'algoritmo sopra menzionato.

La seconda idea per quanto apparentemente più complessa, è stata formulata ipotizzando la possibilità che questo approccio possa in realtà rivelarsi più semplice.

Sulla prima idea si basano le seguenti ipotesi di algoritmo:

- Individuare la cellula, analizzando l'istogramma del colore, realizzando un metodo di sogliatura adattivo
- Applicare una sogliatura e individuare forme circolari o ellittiche corrispondenti al leucocita sfruttando tecniche di pattern recognition (RANSAC e trasformata di Hough) ed edge detection (Canny).

Mentre sulla seconda le seguenti ipotesi:

- Identificare singolarmente il background e le altre fonti di rumore (come altre cellule) mediante sogliatura, sommarle (logicamente) tra loro in modo da ottenere la maschera complementare, che sarà invertita (logicamente) per ottenere la cellula.
- Eliminare il background mediante sogliatura manuale e successivamente applicare Otsu

Naturalmente in ognuna di queste ipotesi, è prevista la possibilità di effettuare conversioni di spazi di colore e operazioni morfologiche di rifinitura quando necessario.

12.2 Algoritmi Proposti

Di seguito descriverò nel dettaglio gli algoritmi accennati in precedenza, mostrando come ogni algoritmo lavori nei passi intermedi.

12.2.1 Estrazione Con Sogliatura Manuale

Questo algoritmo, cerca di sfruttare lo stesso principio dell'algoritmo usato da [8] basandosi sui minimi dell'intero istogramma e si sviluppa nei seguenti step:

1. Conversione da RGB a scala di grigi

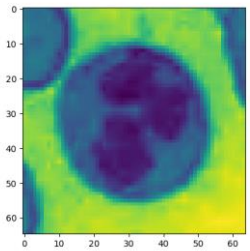

Figura 12.2 ROI dopo la conversione in scala di grigi.

2. Per eliminare eventuali valori alti o bassi che potrebbero falsare il risultato, l'istogramma del colore viene leggermente tagliato sia all'inizio che alla fine (empiricamente ho scelto di tagliare 3 valori all'inizio e 4 alla fine)
3. Si selezionano i minimi dell'istogramma; trattandosi dell'intero istogramma, i minimi potrebbero essere maggiori di uno.
4. Tra i minimi, viene selezionato il primo, cioè quello che si trova più a sinistra dell'istogramma

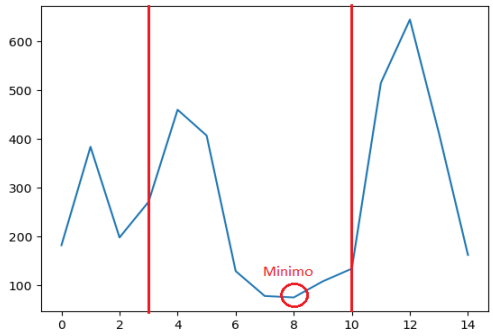

Figura 12.3 Selezione del threshold dall'istogramma del colore.

5. Eliminando tutti i valori maggiori della soglia si ottiene la maschera binaria

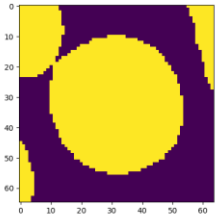

Figura 12.4 Immagine binaria risultante dall'applicazione della soglia

6. Si effettua un’apertura morfologica con disco di 18 pixel

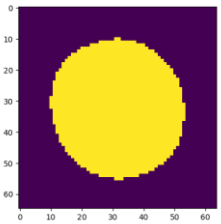

Figura 12.5 Risultato dell'opening

7. Viene selezionata porzione del ROI corrispondente alla maschera ottenuta

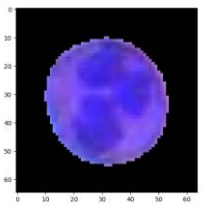

Figura 12.6 Risultato della sovrapposizione tra maschera e immagine originale

Questo algoritmo, risulta molto valido in presenza leucociti con una forma tondeggiante e regolare, ha una buona precisione nell'estrazione, tuttavia genera molti errori come nei seguenti casi:

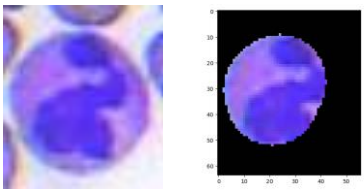

Figura 12.7 Immagine originale (a sinistra), output dell'algoritmo (a destra)

In questo caso è evidente come l'algoritmo abbia tagliato una buona parte di immagine

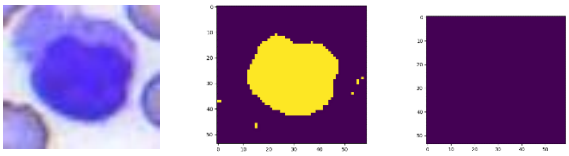

Figura 12.8 Immagine originale (a sinistra), maschera binaria (al centro), apertura morfologica (a destra)

Mentre in questo caso la maschera viene totalmente eliminata durante l'operazione di apertura. Si potrebbe pensare di ridurre la dimensione del disco utilizzato nell'apertura morfologica, in realtà la dimensione scelta è tale da garantire la massima precisione sul dataset a disposizione.

Perciò risulta inefficiente in caso di leucociti irregolari o con il citoplasma poco marcato.

12.2.2 Estrazione Basata Su Circle Matching

L'algoritmo effettua una sogliatura iniziale, per eliminare la maggior parte del rumore, poi attraverso algoritmi di pattern recognition individua delle forme circolari nell'immagine, al fine di ottenere una maschera in grado di riconoscere anche cellule di dimensioni inferiori con un buon grado di precisione. Ho dotato l'algoritmo, della possibilità di scegliere l'algoritmo di fitting di forme circolari, utilizzando sia RANSAC, sia la trasformata di Hough, già menzionati in precedenza. Entrambi sono algoritmi iterativi e ho deciso di metterli a confronto

all’interno dello stesso algoritmo, in quanto dalla letteratura non sono emersi dettagli sufficienti a preferire uno piuttosto che l’altro. L’algoritmo si sviluppa nei seguenti step:

1. Conversione da RGB a scala di grigi

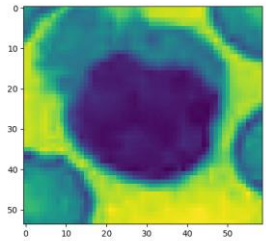

Figura 12.9 ROI convertito in scala di grigi

2. Anche in questo caso è necessario tagliare la parte iniziale e finale dell’istogramma, effettuando un taglio del 10% della lunghezza totale dell’istogramma in egual modo da entrambe le parti.
3. In maniera analoga alle precedenti si considera il primo minimo individuato
4. Dall’immagine in scala di grigi si eliminano tutti i pixel maggiori della soglia
5. Si applica l’algoritmo di Canny con varianza a 1.5, che restituisce la maschera dei bordi.

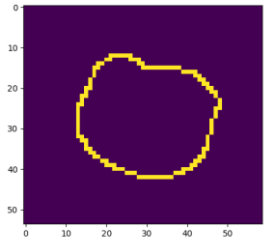

Figura 12.10 Contorni ottenuti con l'algoritmo di Canny.

6. A questo punto se si utilizza Hough:
    a. Si applica la trasformata di Hough alla maschera dei contorni, passandogli la lista dei raggi dei possibili cerchi da ricercare. Sono stati utilizzati raggi in un range da 20 a 50 pixel con variazioni di 2 pixel e il numero di *peaks* (8) cioè il numero di risultati da restituire
    b. La trasformata restituisce le coordinate dei centri e i raggi dei migliori 8 risultati e si effettua la media delle coordinate e dei raggi, in modo da ottenere un solo cerchio.

c. Vengono calcolati i pixel che faranno parte del perimetro del cerchio individuato

7. Altrimenti se si utilizza RANSAC:
   a. Dalla maschera dei bordi si ottengono le coordinate di tutti i pixel corrispondenti ai contorni
   b. Si esegue RANSAC, passandogli il modello matematico del cerchio, le coordinate dei punti, il numero di punti minimo per eseguire il *fitting* (3 punti per il cerchio), la distanza massima dal modello per considerare un punto un *inlier* (impostato a 2) e il numero di iterazioni massime per ogni modello testato (1000 iterazioni)
   c. RANSAC restituisce gli *inliers* e il miglior modello "fittato". Il modello corrisponde all'equazione di un cerchio, perciò vengono calcolati i pixel che fanno parte del perimetro del miglior cerchio trovato.
8. il perimetro del cerchio calcolato con uno dei due algoritmi, potrebbe cadere fuori dall'immagine, perciò vengono eliminati i punti non rientranti nell'immagine
9. Viene disegnato il cerchio.

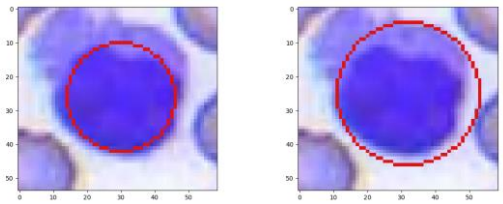

Figura 12.11Modello circolare individuato dall'algoritmo basato su RANSAC (a sinistra) e trasformata di Hough (a destra).

Come è possibile notare sia RANSAC che Hough eliminano parte della cellula anche se in RANSAC questo effetto è più accentuato; in caso di cellule, più regolari invece i due metodi si comportano in modo simile, anche se in questo caso RANSAC è stato un più preciso di Hough.

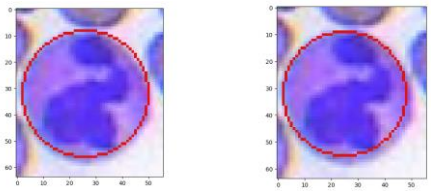

Figura 12.12 Confronto tra RANSAC (a sinistra) e trasformata di Hough (a destra) per cellule tondeggianti.

Durante i test RANSAC pur essendo molto rigido nella selezione del modello, ha una precisione maggiore di Hough anche se risulta più lento; in quanto necessita di molte iterazioni per avere un grado di precisione accettabile. Tuttavia in cellula non regolari, anche questo algoritmo, genera molte imprecisioni.

Sono stati effettuati test con un modello ellittico, in quanto più adattabile a cellule irregolari, ma è stato scartato a causa della mole di tempo necessaria per la creazione e il fitting dei modelli, in quanto il modello ellittico richiede un numero maggiore di parametri rispetto al cerchio[1].

È possibile notare che in questo caso è stato effettuato un taglio differente per l'istogramma. Questa scelta, così come tutti gli altri parametri fissi, sono stati affinati durante i test, al fine di ottenere dall'algoritmo, la miglior prestazione con il maggior numero di esempi.

[1] È stato testato anche la combinazione del metodo di Otsu con l'algoritmo di RANSAC, ma visti i risultati mediocri, si è preferito non descriverlo in questo lavoro.

12.2.3 Estrazione Basata Su Rimozione Del Background e Degli Oggetti Estranei

Nel seguente algoritmo sono stati utilizzati due metodi, che ho sviluppato in maniera specifica per **identificare il background** e gli **altri oggetti estranei** presenti nelle immagini che ho utilizzato singolarmente anche in altri algoritmi, perciò ritengo opportuno descriverli separatamente, per poi illustrare gli step dell'intero algoritmo.

12.2.3.1 Metodo Di Rilevamento Del Background

Per eseguire la rimozione del background vengono eseguiti i seguenti step:

1. L'immagine viene convertita da RGB a scala di grigi
2. Si genera l'istogramma del colore, impostando una suddivisione in 255 livelli di colore (impostando il parametro *bins* [16]). Questo viene effettuato per avere una corretta distribuzione dei colori sull'asse delle ascisse.[2]
3. La soglia viene individuata dal valore che ha il più basso numero di occorrenze nell'istogramma compreso tra due soglie determinate euristicamente (compresi tra 100 e 230)

[2] Vengono considerati 255 colori e non 256, in quanto sperimentalmente si è notato che il valore 1, cioè il valore massimo nella scala di grigi, è assente in tutte le immagini e di conseguenza sono sempre presenti al più 255 colori.

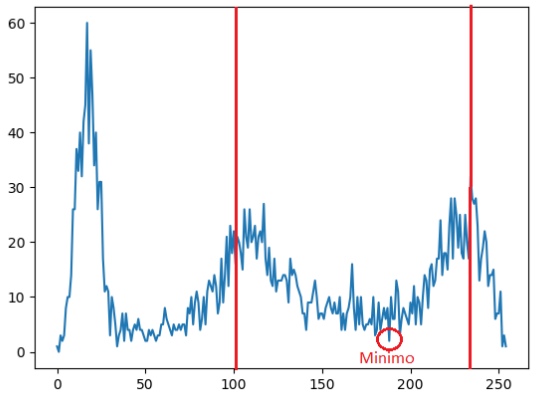

Figura 12.13 Thresholding dell'istogramma del colore per il rilevamento del background, nel range compreso tra 100 e 230.

4. Si crea la maschera binaria, mantenendo solo i pixel minori della soglia trovata

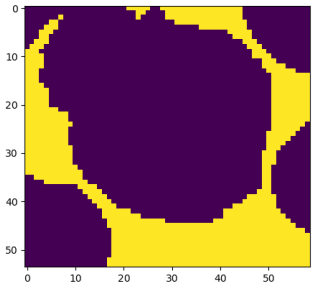

Figura 12.14 Maschera binaria del background.

5. A questo punto è possibile identificare il background sovrapponendo la maschera all'immagine originaria

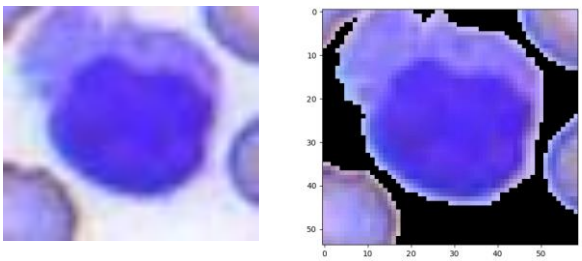

Figura 12.15 Immagine originale (a sinistra), immagine con background rimosso (a destra).

12.2.3.2Metodo Di Rilevamento Degli Oggetti Estranei

Per individuare gli oggetti estranei all'interno dell'immagine, vengono eseguiti i seguenti step:

1. Si estrae il livello Blue dall'immagine RGB
2. Si converte l'immagine da RGB a scala di grigi
3. Si generà l'istogramma del colore del canale Blu, impostando una suddivisione su 100 livelli (sperimentalmente si è visto che questa suddivisione, fornisce risultati migliori).
4. Per ogni valore dell'istogramma, si effettua, partendo dall'ultimo e procedendo a ritroso, la somma parziale delle occorrenze
5. La soglia cercata, corrisponde al valore la cui somma parziale corrisponde al 70% delle occorrenze totali dell'istogramma.
6. Elimino dal canale Blue tutti i pixel maggiori della soglia.
7. I restanti pixel corrispondono alla maschera binaria degli oggetti estranei.

Questo algoritmo è frutto di pura sperimentazione, i parametri fissi utilizzati, sono stati determinati sulla base dei test effettuati. Il metodo di ricerca della soglia, su basa sulla struttura dell'istogramma del canale Blue. Infatti in tutte le immagini, la maggior parte delle occorrenze si distribuisce nella parte finale dell'istogramma.

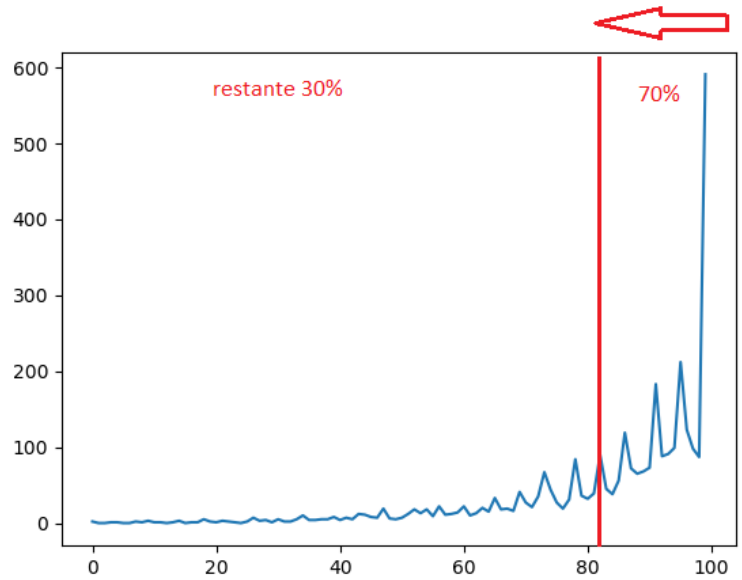

Figura 12.16 Rilevazione della soglia ottimale. La maggior parte delle occorrenze distribuisce per valori alti di blu (sulla parte destra).

Quindi per raggiungere il 70% delle occorrenze totali, è necessario sommarle partendo dalla fine. Questo metodo tuttavia, presenta un certo grado di errore nell'individuazione, spesso vengono selezionate anche porzioni di globulo bianco.

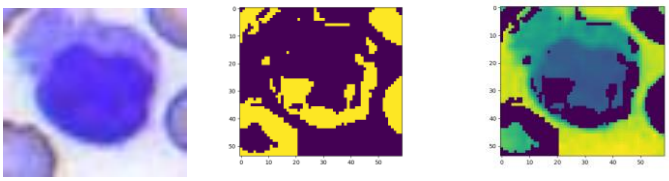

Figura 12.17 Immagine originale (a sinistra), maschera binaria (al centro), immagine in scala di grigi con la maschera applicata (a destra).

12.2.3.3Algoritmo

L'algoritmo basato di **rimozione del background e degli oggetti estranei**, si sviluppa nei seguenti step:

Figura 12.18 ROI di partenza.

1. Si identifica la maschera del background con il metodo di identificazione del background
2. Si identificala maschera degli oggetti estranei, con il metodo di identificazione degli oggetti estranei
3. Si effettua una somma logica delle due maschere binarie

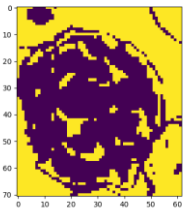

Figura 12.19 Somma logica della maschera del background e degli oggetti estranei.

4. Si inverte la maschera negandola logicamente
5. Vengono rimossi eventuali oggetti di dimensione inferiore a 400 pixel
6. Vengono riempiti eventuali buchi nella maschera con dimensione massima di 200 pixel

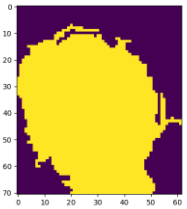

Figura 12.20 Risultato dello step 4,5,6.

7. Si effettua un'apertura morfologica con disco da 8 pixel

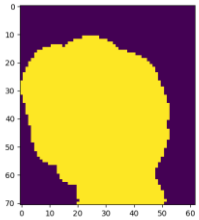

Figura 12.21 Apertura morfologica.

8. La maschera ottenuta viene sovrapposta sull'immagine originale, per ottenere il globulo bianco

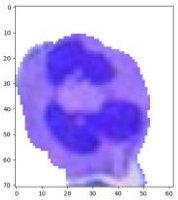

Figura 12.22 Sovrapposizione della maschera all'immagine originale.

Questa tecnica riesce ad eliminare velocemente il rumore, tuttavia non risulta abbastanza precisa nel rilevamento dei contorni.

12.2.4 Estrazione Basata Su Rimozione Del Background e Sogliatura Di Otsu

Nel seguente algoritmo, su utilizzerà il metodo di identificazione del background visto in precedenza, combinato con il metodo di Otsu. L'idea è quella di migliorare la sogliatura di Otsu eliminando in anticipo dall'immagine il background, che è elemento di disturbo, per poi rifinire la maschera con operazioni morfologiche.

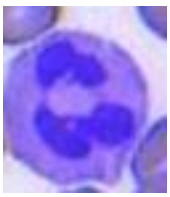

Figura 12.23 ROI di partenza.

1. Identifico la maschera del background (con il **metodo di identificazione del background**)

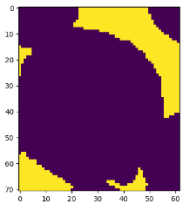

Figura 12.24 Maschera del background

2. Elimino il background dall'immagine RGB sfruttando la maschera appena creata
3. Estraggo il canale Blu dall'immagine RGB

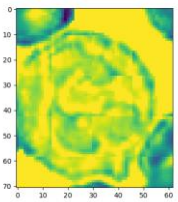

Figura 12.25 Canale Blu dell'immagine RGB.

4. Applico l filtro gaussiano al canale Blu, per la rimozione del rumore

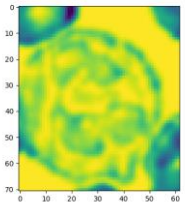

Figura 12.26 Applicazione del filtro Gaussiano.

5. Calcolo la soglia con il metodo di Otsu
6. Applico la soglia T al canale Blu filtrato precedentemente (con la gaussiana), tagliando tutti i valori maggiori di T e ottenendo la maschera binaria

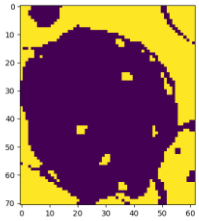

Figura 12.27 Mashera risultante col metodo di Otsu.

7. Applico una chiusura morfologica (con l'elemento strutturale di default, un quadrato con connettività di valore 1)
8. Riempio i buchi maggiori di 400 pixel nella maschera
9. Effettuo una erosione con disco di 4 pixel

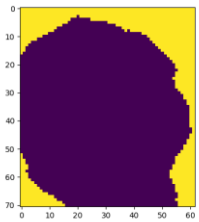

Figura 12.28 Risultato dopo lo step 7,8,9.

10. Inverto la maschera negandola logicamente
11. Applico un'apertura con un disco da 10 pixel

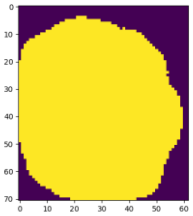

Figura 12.29 Negazione logica della maschera e apertura.

12. Applico la maschera all'immagine originale per estrarre il leucocita

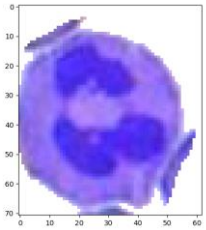

Figura 12.30 Risultato finale dopo l'applicazione della maschera.

Come è evidente dall'immagine risultate, è ancora presente del rumore, dovuto a cellule adiacenti al leucocita, tuttavia il risultato è ottimo. Durante i test questo algoritmo, si è rivelato uno dei più stabili, pur essendo meno preciso di altri, ottiene risultati simili anche per cellule con differenti forme e dimensioni, quindi si propone come ottimo candidato.[3]

12.3 CONFRONTO

Dopo aver testato gli algoritmi singolarmente e in varie versioni, ho effettuato dei test, per confrontarli tra loro, in modo da stabilire quale fosse il miglior algoritmo da utilizzare nella fase di estrazione. Per la valutazione ho sottoposto gli algoritmi alle stesse immagini di input. Di seguito sono messi a confronto con cellule rappresentative. Si è cercato in questi esempi di mostrare i punti di forza e di debolezza di ogni algoritmo.

ROI	Sogliatura Manuale	RANSAC	Rimozione sfondo e oggetti estranei	Rimozione Sfondo e metodo di Otsu

[3] Sono stati effettuati test combinando il metodo di Otsu e la rimozione degli oggetti estranei, ma i risultati non sono stati sufficientemente rilevanti da essere descritti nel lavoro di tesi.

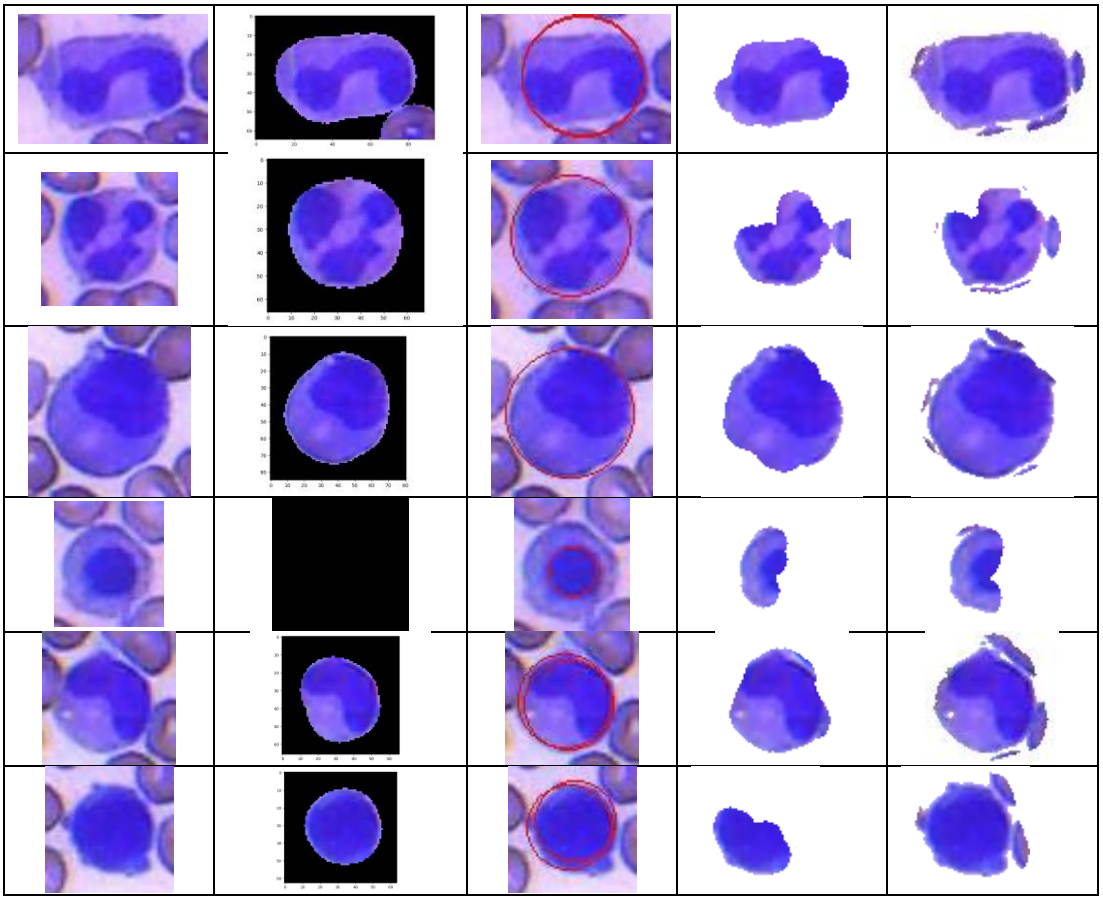

Figura 12.31 Tabella di confronto.

Il metodo con sogliatura manuale come già evidenziato in precedenza, risulta molto preciso dell'individuazione dei contorni della cellula soprattutto in caso di cellule con contorni regolari. In caso di cellule piccole (come nel caso della maggior parte dei *linfociti piccoli*), la soglia è molto restrittiva e ne impedisce l'individuazione.

Il metodo basato su RANSAC, riesce sempre ad individuare una regione corrispondente alla cellula, ma risulta restrittivo per quanto riguarda la forma, che è vincolata alla figura del cerchio. Sono stati effettuati test con modelli ellittici, ma sono risultati inefficienti, in quanto il maggior numero di parametri, rallenta molto l'esecuzion. Il tempo impiegato seppur con risultati migliori, non è accettabile considerando le migliaia di cellule presenti in un vetrino. Sono state effettuate altre sperimentazioni, come combinazione tra RANSAC e metodo di Otsu, ma i risultati non sono stati soddisfacenti.

Il metodo basato sulla rimozione degli oggetti estranei e rimozione del background, si adatta meglio alla variabilità delle immagini, ma risulta instabile nei risultati, producendo molti output, con grandi porzioni di cellula mancanti. Anche questo algoritmo è risultato instabile su tutto il dataset.

Il metodo basato sulla rimozione del background e sul metodo di Otsu, ha un adattamento simile al metodo precedente, anche se tende ad eccedere dell'individuazione dei contorni della cellula includendo del rumore nelle zone periferiche. Tuttavia quest'ultimo metodo, su tutto il dataset, ha mostrato molta più stabilità alla variazione dell'input. La stabilità, intesa come capacità di fornire l'output, in maniera coerente al variare dell'input, era il principale problema da risolvere negli algoritmi testati. Perciò si è deciso di utilizzare questo algoritmo come base, per l'algoritmo di estrazione finale.

12.4 Algoritmo Finale Di Estrazione

L'algoritmo finale è stato ottenuto da un processo di sviluppo e ottimizzazione sperimentale sul dataset, perciò sono stati adottati accorgimenti tecnici, per prevenire possibili errori d'esecuzione. L'**algoritmo di estrazione dei leucociti** si sviluppa nei seguenti step:

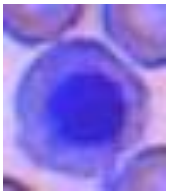

Figura 12.32 ROI del leucocita non riconosciuto in modo corretto da nessun algoritmo precedente, presente nella Figura 12.31.

1. Si individua e si rimuove il background dall'immagine RGB.

Figura 12.33 Maschera del background.

2. Si estrae il canale Blu dall'immagine RGB

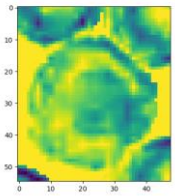

Figura 12.34 Canale Blu dell'immagine RGB.

3. Si applica un filtro gaussiano per l'eliminazione del rumore

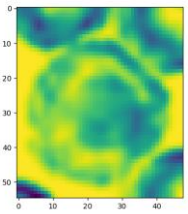

Figura 12.35 Applicazione del filtro Gaussiano al canale Blu.

4. A questo punto si effettua un controllo sulla presenza di una variazione di colore; infatti se l'immagine è a tinta unita o presenta bassissime variazioni di colore, non rispetta le precondizioni del metodo di Otsu
5. Se sono rispettate le precondizioni, si applica il metodo di Otsu, all'immagine filtrata, evidenziando i valori minori della soglia, quindi quelli più scuri.

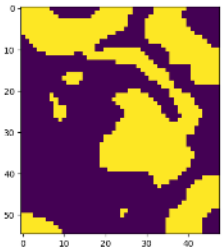

Figura 12.36 Maschera binaria dopo l'applicazione del metodo di Otsu.

6. In questa maniera, si evidenziano le zone con una più forte tonalità di blu, compreso il nucleo e i bordi delle cellule circostanti. Ora si somma logicamente la maschera del background calcolata in precedenza a quella ottenuta con Otsu

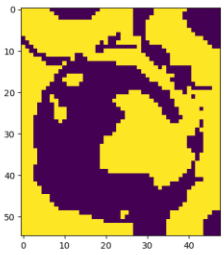

Figura 12.37 Somma logica della Figura 12.35 e Figura 12.32.

7. Se consideriamo la maschera inversa, cioè quella non evidenziata, si può notare come questo procedimento, sia riuscito ad evidenziare il citoplasma del leucocita, tralasciando tutto il plasma, e quasi totalmente le cellule estranee. Il nucleo, tuttavia resta evidenziato; per eliminarlo adottiamo uno stratagemma: nel ROI, se è contenuto un leucocita, questa indipendentemente dalla sua grandezza, sarà approssimativamente centrata rispetto al ROI stesso. Perciò ci basterà eliminare dalla maschera un oggetto sufficientemente piccolo da rimuovere buona parte del nucleo, ma non troppo grande da superare la grandezza della cellula. La dimensione ottimale, si è ottenuta disegnando al centro dell'immagine un disco di raggio pari a un terzo della larghezza dell'immagine (si può utilizzare anche l'altezza, in quanto le immagini sono quasi quadrate). Tale disco si andrà a sottrarre dalla maschera corrente, ottenendo il seguente risultato

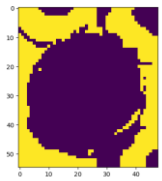

Figura 12.38 Risultato dello step 7.

8. Ora che la forma del leucocita è più evidente, si va a compattare la maschera effettuando una chiusura morfologica
9. Si riempiono i buchi con dimensioni massima di 400 pixel
10. Si effettua un'erosione con disco da 5 pixel

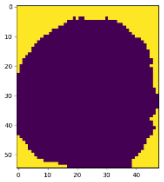

Figura 12.39 Risultato dello step 8, 9, 10.

11. Si inverte la maschera
12. Si effettua una chiusura con disco da 10 pixel
13. E si riempiono eventuali piccoli buchi ancora presenti (con l'elemento strutturale di default, un quadrato con connettività di valore 1)

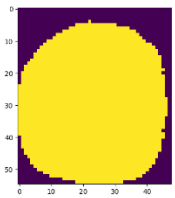

Figura 12.40 Risultato step 11,12,13.

14. Si verifica se la maschera copre l'intera immagine o è assente, in modo da evitare, immagini non estratte o immagini bianche
15. Si applica la machera all'immagine originale

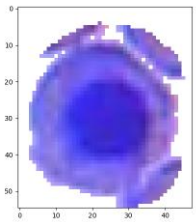

Figura 12.41 Immagine finale, non riconosciuta dagli algoritmi della Figura 12.31.

Algoritmo proposto è più stabile alla variazione delle immagini di input e anche se presenta mancanza di precisione nella determinazione dei bordi, è in grado di riconoscere cellule di diversa dimensione e forma.

Si sono fatte varie considerazioni anche con la valutazione di uno specialista ematologo, sulla ripercussione dovuta all'errore di estrazione del bordo della cellula nel calcolo delle feature. Tale errore si può considerare ininfluente, in quanto si ripresenta su tutte le cellule nelle stesse modalità e anche se intaccasse valori delle features, nella sua stabilità e "coerenza", non andrebbe a falsare l'estrazione delle features.

13 SELEZIONE DELLE FEATURES

In letteratura sono già state utilizzate diverse tipologie di features geometriche, basate su tessitura o basate su colore. [8]. In base allo studio condotto da [8], sono state selezionate 7 features, tra 33 presenti in letteratura, utilizzando l'algoritmo *InfoGainAttributeEval* che fa uso della tecnica dell'*information gain* (IG), per determinare la rilevanza di un attributo rispetto agli altri in un problema computazionale.

Le 7 features selezionate da [8] sono:

1. Valore di intensità del canale Blu più frequente nel citoplasma
2. Valore di intensità del canale Hue più frequente nel citoplasma
3. Numero di lobi
4. Solidità del nucleo
5. Area della cellula
6. Rapporto delle aree del citoplasma e del nucleo
7. Valore medio di intensità del grigio della cellula

Tranne la 3, l'estrazione di queste feature non è particolarmente complessa da essere descritta in dettaglio. Per la 1 la 2 e la 7 viene analizzato l'istogramma del colore, per la 4 viene calcolata l'area dell'inviluppo convesso (*convex hull area*) sfruttando l'algoritmo proposto da [14], per la 5 viene calcolata l'area dei pixel della cellula e per la 6 viene effettua un'operazione analoga.

Nelle prossime sezioni, descriverò i metodi utilizzati per l'estrazione del nucleo e del citoplasma dal leucocita estratto nella fase precedente e descriverò l'algoritmo del calcolo dell'unica feature non banale da calcolare cioè la 3 ovvero il numero di lobi.

13.1 Individuazione Del Nucleo e Del Citoplasma

Come algoritmo di estrazione del nucleo e del citoplasma si è pensato di implementare quello proposto da [8], tuttavia, non ha fornito risultati all'altezza di quelli mostrati, perciò è stato rielaborato nel seguente algoritmo.

1. Si converte la cellula estratta da RGB a HSV e si seleziona il canale S

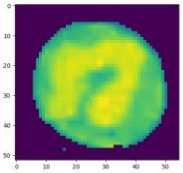

Figura 13.1 ROI convertito in HSV.

2. Si applica il filtro gaussiano allo scopo di smussare l'istogramma del colore
3. Si impostano i pixel con valore inferiore a 0.1 a 1. In questo modo lo sfondo della cellula diventa scuro, cioè molto più vicino ai valori dei pixel presenti nel nucleo. In questo modo si ottiene una distinzione più netta tra sfondo e citoplasma.

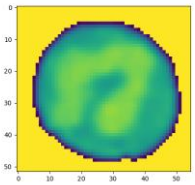

Figura 13.2 Maschera dello sfondo.

4. Si calcola la soglia T con Otsu, e viene applicata evidenziando i valori maggiori di T.

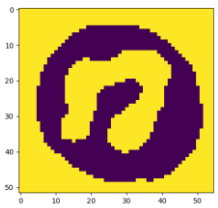

Figura 13.3 Applicazione metodo di Otsu a Figura 13.2.

5. A questo punto si eliminano le componenti connesse con area massima di 95 pixel. Questo processo, rimuovere eventuale rumore dalla maschera
6. Ora basta invertire logicamente la maschera, per ottenere la maschera del citoplasma

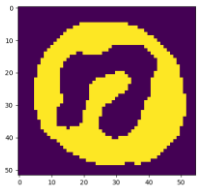

Figura 13.4 Maschera del citoplasma.

7. Mentre per ottenere la maschera del nucleo si rimuovono dalla maschera tutti i pixel dell'immagine prodotta dallo step 5 che appartengono allo sfondo cioè quelli che nella gaussiana sono stati impostati con valore 1.

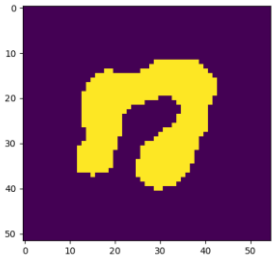

Figura 13.5 Maschera del nucleo.

Durante i test, questi metodi si sono rivelati stabili su tutto il dataset senza mostrare particolari cali di precisione.

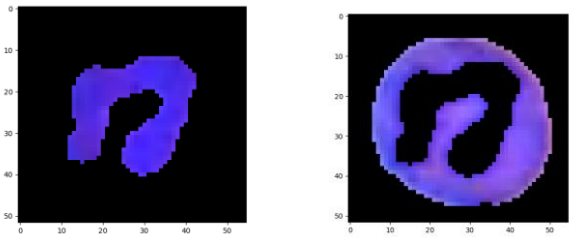

Figura 13.6 estrazione del nucleo (a sinistra) e del citoplasma (a destra).

14 ESTRAZIONE DELLE FEATURES

In machine learning, pattern recognition e image processing, l'estrazione delle features, consiste nella costruzione di dati significativi (features), a partire da un insieme di dati misurati, allo scopo di facilitare l'apprendimento e la generalizzazione del problema che si sta affrontando, fornendo in molti casi risultati migliori delle interpretazioni umane.

Quando i dati di input sono troppi e ridondanti, è necessario trasformarli in un insieme più ridotto di dati con maggior contenuto informativo, chiamato **features**. Questa tecnica è fondamentale nel riconoscimento di oggetti all'interno di immagini; infatti all'interno di un'immagine, sono presenti molti pixel che non appartengono all'oggetto che si intende individuare e di conseguenza molti pixel non hanno contenuto informativo. Inoltre un'immagine anche se piccola può contenere migliaia di pixel e la loro elaborazione risulta inutile e lenta. Una **feature** è un dato più complesso che deriva dall'insieme iniziale di dati e che fornisce un'informazione rilevante per risolvere un determinato problema computazionale.

14.1 Analisi Dei Lobi

Il numero dei lobi è particolarmente caratterizzante per i neutrofili e gli eosinofili, che contengono per l'appunto nuclei multi-lobati, mentre monociti, linfociti e basofili, presentano per la maggior parte nuclei lobati o bi-lobati. In letteratura sono stati proposti vari metodi per effettuare la conta dei lobi [8], per lo più basate su operazioni morfologiche. Lo stato dell'arte in questo ambito, sono i metodi basati su grafi e scheletro morfologico (**skeletonize**) come il metodo proposto da [3].

L'algoritmo di conta dei lobi realizzato si articola in 8 step:

1. Calcolo dello scheletro morfologico
2. Calcolo dello scheletro delle distanze
3. Segmentazione dello scheletro
4. Rimozione dei segmenti troppo piccoli
5. Calcolo le distanze di ogni segmento
6. Verifica delle condizioni di taglio su ogni segmento
7. Labeling e conta dei segmenti

Nei seguenti paragrafi verranno descritti i singoli passaggi.

14.1.1 Calcolo Dello Scheletro Morfologico

Il calcolo dello scheletro morfologico, viene effettuato con la tecnica dello skeletonize che fa uso del thinning, come spiegato in precedenza. Dallo scheletro. Vengono individuati le seguenti tipologie di punti:

- **end-point**: punti che non hanno altre diramazioni
- **branch-point**: punti da cui si diramano altri segmenti

Questi punti costituiscono i possibili estremi di un segmento e verranno utilizzati nella segmentazione. È stato necessario implementare un metodo ad hoc, per il rilevamento dei branch-point e degli end-point, in quanto non presente nelle librerie utilizzate.

Per l'identificazione, si è fatto uso dell'algoritmo **hit-and-miss** e sono stati introdotti manualmente gli elementi strutturali rappresentanti tutti i possibili branch-point ed end-point.

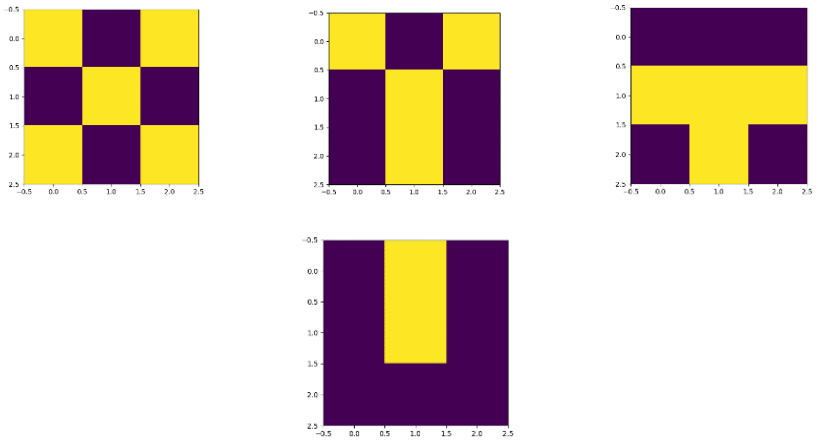

Figura 14.1 Elementi strutturali dei possibili branch-point (in alto), a X (a sinistra), a Y (al centro), a T (a destra). L'unico end-point (in basso). Nell'algoritmo, sono utilizzate anche tutte le loro possibili rotazioni.

Durante i test è emerso che gli elementi strutturali utilizzati in letteratura, mancano di una struttura utile al rilevamento di un particolare branch-point che di conseguenza è stata inserita nell'algoritmo.

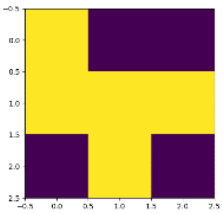

Figura 14.2 Nuovo branch-point individuato.

14.1.2 Calcolo Dello Scheletro Delle Distanze

Il calcolo delle distanze, viene effettuato applicando la trasformata *medial axis*, alla maschera del nucleo, che viene avvalorata con i valori delle distanze di ogni pixel dal bordo. La maschera delle distanze viene sovrapposta allo scheletro ottenuto nella fase precedente, per ottenere lo scheletro delle distanze.

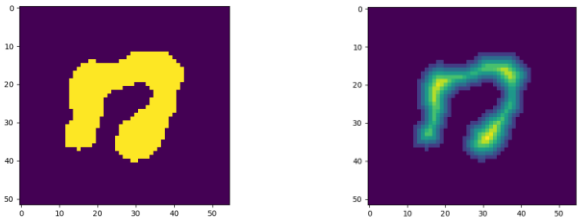

Figura 14.1 Maschera del nucleo (a sinistra), applicazione della trasformata medial axis (a destra).

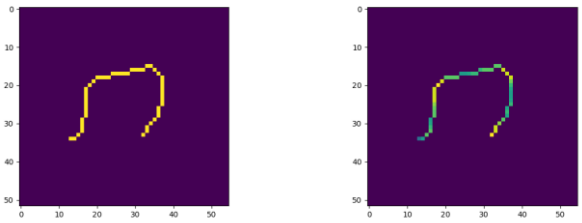

Figura 14.2 Maschera dello scheletro (a sinistra), scheletro delle distanze (a destra).

14.1.3 Segmentazione Dello Scheletro

Anche la segmentazione dello scheletro, è stata implementa interamente ad hoc. Per ogni punto appartenente agli ep e bp, si va a verificare la presenza di un percorso nello scheletro.

Il percorso viene calcolato partendo da un ep o bp e andando ad analizzare tutti i pixel adiacenti a quello corrente. Ogni pixel analizzato, viene "consumato dall'algoritmo", che lo memorizza temporaneamente come possibile percorso. Quando l'analisi raggiunge un altro punto appartenente agli ep o bp, il percorso trovato viene rimosso dallo scheletro e salvato separatamente. Se nel percorso vi erano degli ep, questi sono rimossi dallo scheletro, mentre ciò non avviene per i bp, perché potrebbero essere connessi ad un altro percorso. Quest'operazione viene effettuata iterativamente, per ogni punto ep o bp dello

scheletro. Il ciclo termina quando tutti i bp, non sono connessi a nessun segmento. Per ogni segmento individuato, viene restituita la maschera corrispondente al segmento, la lunghezza del segmento, e le coordinate di ogni punto del segmento, nell'immagine.

14.1.4 Rimozione Di Piccoli Segmenti

Dalla lista di segmenti ottenuti, vengono eliminati quelli con lunghezza inferiore a 4 pixel. Se viene eliminato un segmento, è necessario ricalcolare gli ep e i bp, e rifare la segmentazione, in quanto il segmento eliminato dallo scheletro, converte uno o due bp rimanenti in ep e ciò potrebbe compromettere la segmentazione effettuata. Questo procedimento viene ripetuto fino a quando non vengono effettuate ulteriori rimozioni.

14.1.5 Calcolo Delle Distanze Di Ogni Segmento

Lo scheletro delle distanze viene utilizzato per avvalorare le distanze di ogni segmento.

14.1.6 Verifica Delle Condizioni Di Taglio

Se un segmento è troppo lungo, esso viene tagliato in quanto potrebbe contenere più lobi. Si va ad analizzare il grafico delle distanze di ogni segmento e se si verificano le condizioni di taglio, il segmento viene spezzato.

Le condizioni di taglio, si verificano in presenza di particolari picchi, che vengono identificati nel seguente modo:

1. Vengono eliminati i picchi agli estremi dell'istogramma, tagliando una porzione pari al 15 % della sua lunghezza, sia all'inizio, sia alla fine.
2. Si individua il picco minimo
3. Per ogni minimo, si individuano i picchi massimi in un intervallo di 7 punti a destra e 7 punti a sinistra del picco minimo.
4. A questo punto **si calcola il rapporto e la differenza tra massimo e minimo, effettuando il taglio solo se il primo è maggiore di 1.5, e il secondo è maggiore di 2**

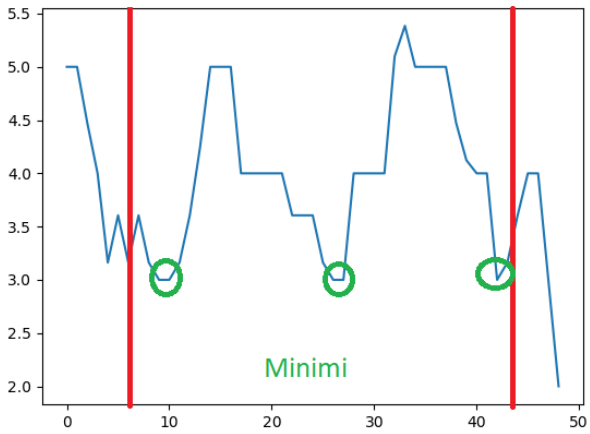

Figura 14.3 Rilevamento dei minimi nell'istogramma delle distanze del segmento.

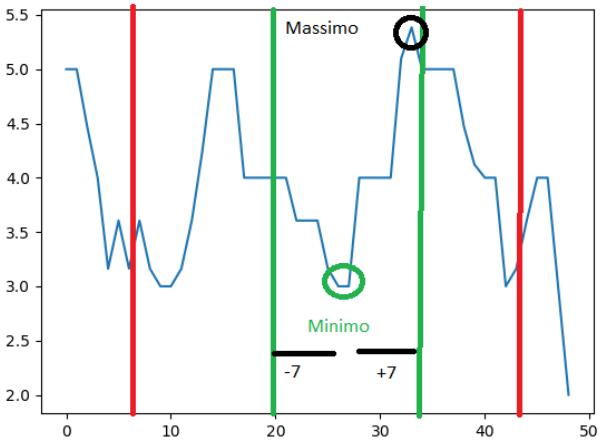

Figura 14.4 Metodo di rilevamento dei massimi per ogni minimo trovato nell'istogramma.

14.1.7 Labeling e Conta Dei Segmenti

Sullo scheletro segmentato, si effettua il labeling, che restituisce il numero di segmenti, presenti nella maschera

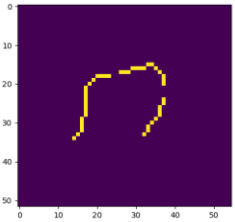

Figura 14.5 Scheletro segmentato, sono stati individuati 3 segmenti.

14.2 Risultati

Come gran parte del progetto, anche questo algoritmo, è stato sviluppato tenendo conto principalmente dei risultati ottenuti sul dataset. L'algoritmo presente in letteratura [8], presentava un altissimo tasso di errore, infatti la quasi totalità dei conteggi erano errati. Questo algoritmo, ha alzato il livello di precisione della conta, ma risulta comunque un considerevole tasso di errore. Gli errori si individuano sia nel principio di funzionamento del processo, sia nella grande diversità dell'input. Infatti in cellule come linfociti e monociti, che hanno nuclei non lobati o a fagiolo, molto spesso vengono individuati 0,1 o 2 lobi. Questo problema è da ricercarsi nella base di funzionamento dello skeletonize; infatti lo skeletonize effettua come già accennato, thinning successivi che vanno ad amplificare le piccole distorsioni del nucleo. Ciò significa che un nucleo apparentemente circolare, può avere uno scheletro a "Y" o "X", con conseguente rilevamento di un numero maggiorie di lobi. In altri casi, lo scheletro è talmente piccolo da essere rimosso, e quindi produrre 0 lobi. Nei ROI analizzati inoltre sono presenti cellule esplose, macchie di colore o altri corpi non riconducibili a cellule, in questi casi l'algoritmo potrebbe comunque tentare di effettuare una conta se viene rilevata una zona riconducibile al nucleo della cellula. Per questi casi sono stati implementati dei controlli appositi che evitano questo tipo di errori nella maggior parte dei casi. Altri problemi sono da riscontrarsi con le cellule lobate; in rari casi, il numero di lobi e sproporzionatamente grande (per esempio 16 o 20 lobi), a causa della particolare forma del nucleo che va a falsare il conteggio.

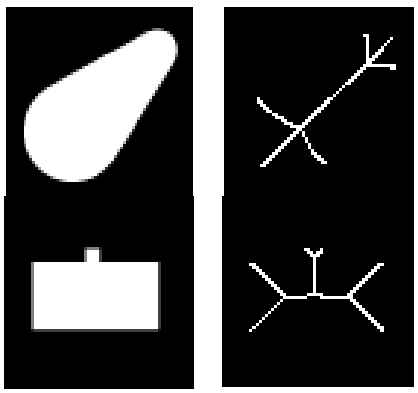

Figura 14.6 Immagini originali (sulla sinistra), applicazione dello skeletonize (sulla destra). Sono evidenti le tipiche ramificazioni che provocano l'errore nella conta dei lobi.

Per bypassare l'errore della conta, ho sviluppato un albero decisionale con lo scopo di discriminare in anticipo, le cellule con nuclei lobati da quelli non

lobati; l'albero analizza prima le altre 6 features e determina se effettuare la conta o meno. Tuttavia non ho avuto modo di testare a fondo la validità di questa soluzione e la si rimanda in possibili sviluppi futuri.

Per analizzare oggettivamente i risultati ottenuti dall'algoritmo è stato necessario selezionare una campione dal dataset. Ho selezionato casualmente un campione di 2400 ROI sui quali ho determinato manualmente il numero di lobi nucleari che ho confrontato con i risultati prodotti dall'algoritmo. Prima di valutare i risultati dell'algoritmo sul campione, ho ritenuto necessario analizzare il campione stesso, per valutarne le caratteristiche e avere un'idea della reale composizione del campione.

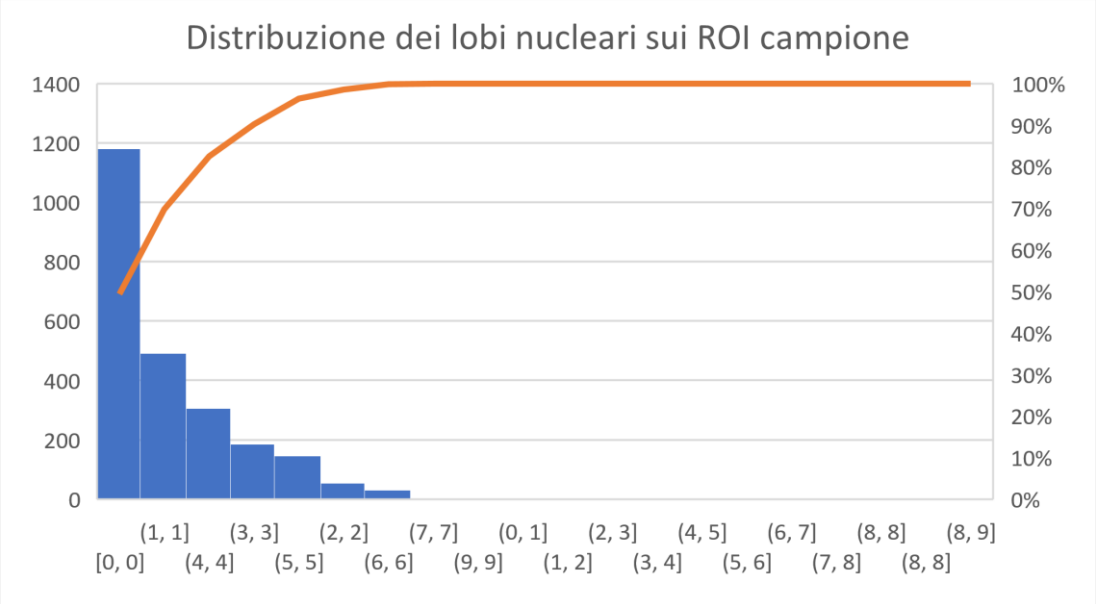

Figura 14.7 Diagramma di Pareto dei lobi nucleari dei ROI campione.

Anche se selezionato casualmente, il campione non risulta uniforme rispetto al numero di lobi nucleari. I ROI con 0 lobi rappresentano la parte più numerosa, occupando il 50% della totalità del campione. Naturalmente cellule senza lobi, non sono cellule, ciò significa che il 50% del campione non è una cellula. Se si aggiungono i ROI con 1, 4, 3 lobi, si supera il 90% del numero di ROI del campione. Quindi ROI con 2 o più di 5 lobi, rappresentano una parte inferiore al 5% del campione. Il fatto che il 50% del campione sia essenzialmente del "rumore" è causato dalla prima parte del sistema generale, cioè quella di estrazione del ROI dal vetrino. Durante l'estrazione infatti, molto spesso vengono generati ROI che non contengono leucociti, tuttavia si è deciso di mantenere anche queste immagini nel dataset, in quanto fanno parte di una

casistica reale, di possibili errori di individuazione. Gli algoritmi delle successive fasi, come l'estrazione delle features, sono stati realizzati tenendo conto della possibile presenza di questi dati che costituiscono rumore, in quanto nonostante siano stati presi i dovuti accorgimenti, la casualità del vetrino è tale da non permetterne la totale eliminazione già dalla prima parte del sistema; perciò si è deciso di considerarli come una classe ad hoc di dati, da riconoscere e gestire separatamente.

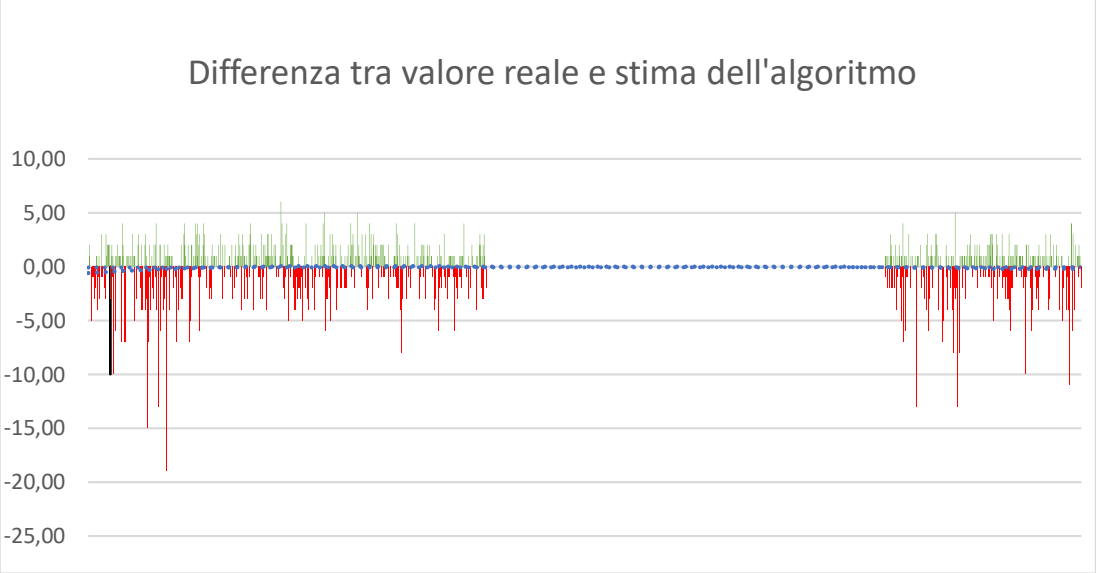

Figura 14.8 istogramma della stima dei lobi. Le barre in rosso identificano una sovrastima dell'algoritmo, le barre in verde una sottostima, mentre la linea tratteggiata indica la tendenza generale

In Figura 14.7 viene mostrata la differenza tra la stima effettuata dall'algoritmo e il numero reale di lobi, presenti nei campioni. Sull'asse delle ascisse ci sono i campioni oggetto della stima, mentre sull'asse delle ordinate la differenza tra il valore reale e il valore stimato. I valori negativi sono in rosso e indicano una sovrastima del valore reale, mentre i valori in verde indicano una sottostima, dove non sono presenti barre la stima è esatta.

La linea tratteggiata indica la tendenza generale dello stimatore, è stato utilizzato uno stimatore polinomiale di grado 6 per avere una stima non troppo semplicistica. Dal grafico si evince come l'algoritmo sia inizialmente leggermente sovrastimante, per poi stabilizzarsi con un andamento neutro. Tuttavia gli sbalzi nella stima sono considerevoli, toccando anche 18 lobi in sovrastima e 7 lobi in sottostima. La maggior parte degli errori rilevanti si attesta intorno a ±5 lobi.

La successiva analisi è stata svolta sulla distribuzione degli errori in base al numero di lobi reali da stimare.

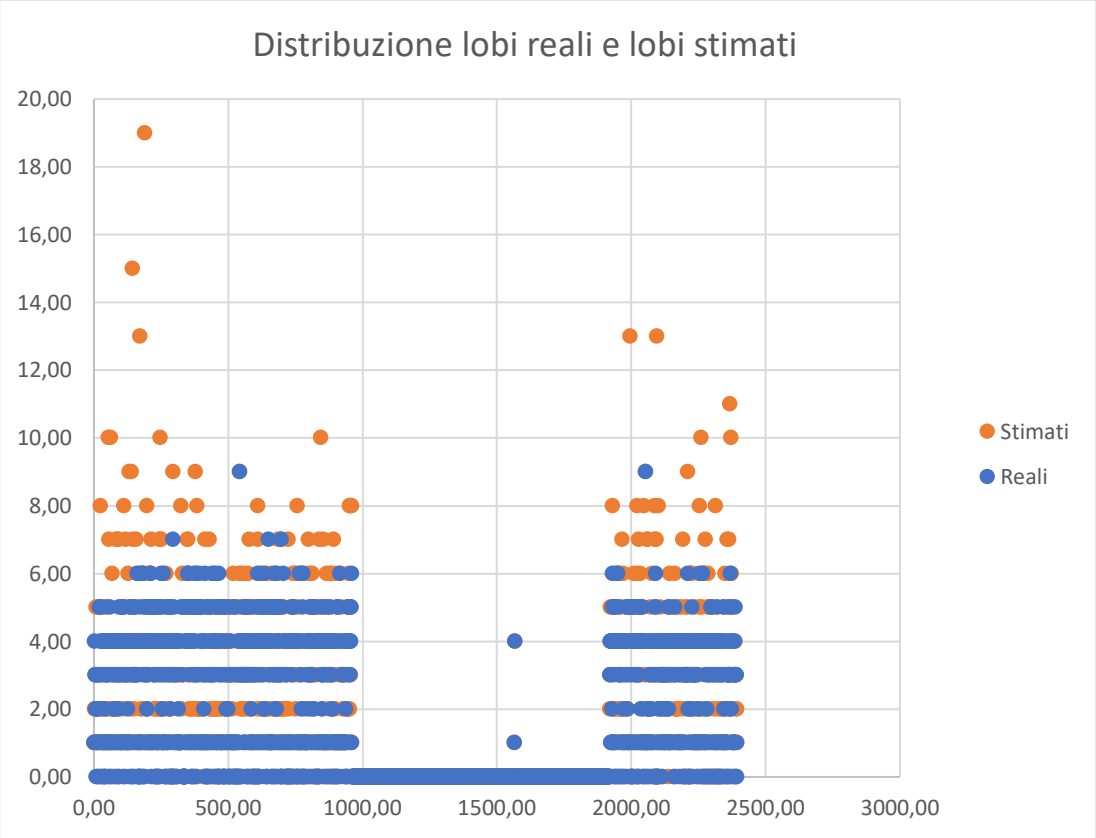

Figura 14.9 Distribuzione dei lobi reali e dei lobi calcolati nel campione

Sull'asse delle ascisse sono presenti i campioni, mentre sull'asse delle ordinate i lobi stimati e reali di ogni campione. In una situazione perfetta la stima e il valore reale coinciderebbero in tutto il campione dalla Figura 14.8 è evidente che ciò si verifica solo per una parte del campione. Infatti fino a valori reali compresi tra 0 e 5 la stima risulta più che buona, coincidendo quasi totalmente con i valori reali, i valori stimati, in alcuni casi stimano un numero di lobi sproporzionato ed in seguito sarà necessario approfondire la tipologia di ROI che generano questo errore.

Dalle analisi qualitative effettuate, non è evidente l'effettiva capacità dell'algoritmo nel restituire l'esatto numero di lobi. Perciò si è ritenuto necessario fornire un'analisi quantitativa dei risultati, per ottenere una panoramica su entrambi i versanti.

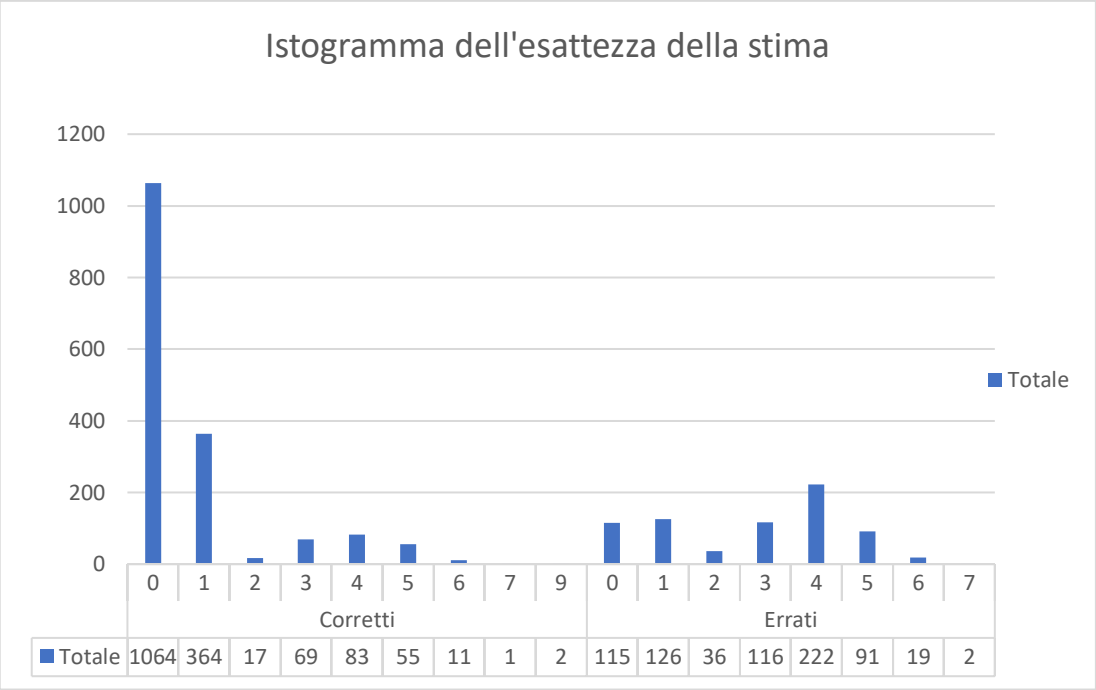

Figura 14.10 Istogramma dell'esattezza della stima. L'istogramma si compone di due sotto-istogrammi: (sulla sinistra), l'istogramma delle stime corrette, in base al numero di lobi, (sulla destra), le stime errate.

Nell'istogramma in Figura 14.10, viene mostrato il numero di valutazioni esatte in base al numero reale di lobi. In questo grafico, non vengono considerate le entità degli errori, ma la semplice correttezza della stima. Nel campione sono state effettuate correttamente 1666 stime contro le 727 errate, con una precisione del 70%. Tuttavia analizzando in dettaglio le valutazioni è evidente, che il 60% delle valutazioni corrette riguarda ROI che non sono cellule. Da un lato evidenzia come le tecniche adoperate per la gestione di ROI "errati" estratti dal vetrino, siano state efficaci, dall'altra, evidenzia le reali capacità dell'algoritmo del conteggio di cellule, specialmente multilobate. Infatti l'algoritmo ottiene un punteggio positivo su ROI con 0 e 1 lobo, mentre ottiene punteggi scarsi per cellule con un numero maggiore di lobi, precisamente:

- 90% per 0 lobi
- 74% per 1 lobo
- 32% per 2 lobi
- 37% per 3 lobi
- 27% per 4 lobi
- 37% per 5 lobi
- 36% per 6 lobi
- 34% per 7 lobi

- 100% per 9 lobi[4]

Dal punto di vista quantitativo la stima esatta dei lobi non è soddisfacente. Tuttavia andando ad analizzare l'entità dell'**errore medio** sul campione, l'errore si attesta su **0.65 lobi** in eccesso o in difetto, ciò significa che anche se la stima non è esatta, nella maggior parte dei casi, viene rilevato un lobo in più o in meno. In base all'analisi svolta, la tecnica della conta dei lobi, allo stato attuale, si propone come una feature più qualitativa che quantitativa e successivamente in questo lavoro, verranno fornite ulteriori motivazioni su questo aspetto, verrà approfondito lo studio sulle reali prestazioni dell'algoritmo della conta dei lobi e saranno esposte ulteriori idee poste allo specifico miglioramento di questo algoritmo.

[4] I punteggi per 6, 7, 9 lobi sono statisticamente irrilevanti rispetto alla dimensione del campione

15 IMPLEMENTAZIONE

Per la realizzazione del progetto, è stata utilizzata la piattaforma Anaconda. Anaconda integra librerie e tool, utili per la realizzazione di software di calcolo scientifico, riunendo librerie open source scritte in Python.

Normalmente per questo genere di implementazioni, vengono utilizzati linguaggi e ambienti specifici, come Matlab, o R. La scelta di usare Python è stata fatta in un'ottica più ampia. Python è un linguaggio ad alto livello orientato a oggetti, utile per sviluppare applicazioni distribuite, scripting, computazione numerica e sistemi di testing. È multi-paradigma, ma la caratteristica principale sono le variabili non tipizzate, l'uso di indentazione per definire le specifiche e un ricco set di operazioni base e librerie standard avanzate. Il controllo sui tipi è forte ma viene eseguito a *runtime* e fa uso di un *garbage collector* per l'ottimizzazione della memoria. Python può essere utilizzato anche come linguaggio lato server, inoltre la sua straordinaria semplicità e l'utilizzo dell'indentazione, rende il codice breve e leggibile, quindi anche manutenibile, oltre a essere versatile ha prestazioni superiori a PHP, Ruby e Java. Nell'implementazione è stato utilizzato Python in versione 3.6.

15.1 Piattaforma Anaconda

Tutto il lavoro di tesi è stato sviluppato in Python, utilizzando la piattaforma Anaconda. Anaconda è un package manger ed enviroment manager, contenente oltre 720 package open source per il calcolo scientifico. È gratuito e supportato da una comunità internazionale e si può definire come un Open Data Science Core.

Figura 15.1 Logo Anaconda.

Di base contiene 150 packages, che possono essere ampliati, scaricandone di più specifici dal repository chiamato Anaconda Cloud. Inoltre contiene componenti, che forniscono alte prestazioni nell'esecuzione di operazioni di calcolo e un IDE lo sviluppo chiamato Spyder per il testing, e il debugging del codice. Spyder è anche un numerical computing enviroment, grazie al supporto nativo ad IPython e alle principali librerie di calcolo come NumPy, SciPy, e matplotlib.

15.2 Ambiente Di Sviluppo

Come ambiente di sviluppo, è stato utilizzato Spyder; cioè l'IDE integrato in Anaconda. Spyder anch'esso open source, fornisce tutti gli strumenti utili allo sviluppo, più alcuni tool specifici, come strumenti di visualizzazione per strutture dati come matrici, vettori e liste, visualizzazione di immagini, possibilità di realizzare grafici e istogrammi, in maniera rapida.

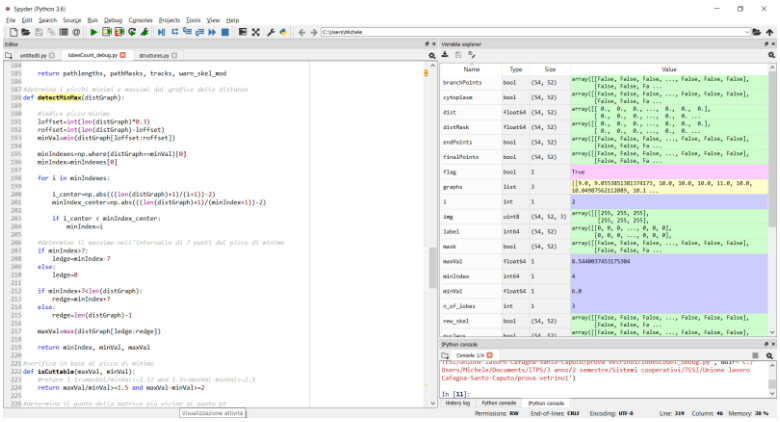

Figura 15.2 Schermata principale dell' IDE Spyder.

15.3 Librerie

15.3.1 Scipy

Scipy (Scientific Computing Tools for Python) è uno dei principali ecosistemi sviluppati per il calcolo scientifico in Python.

Figura 15.3 SciPy Logo.

La struttura di Scipy si basa sullo *Scipy stack*, che è costituito dai principali *core package* tra cui:

- Numpy: è il package fondamentale per il calcolo scientifico, vanta un potente sistema di gestione di matrici n-dimensionali, un a gran numero di funzioni matematiche e di algebra di base avanzate e strumenti di integrazioni con codice C/C++ e Fortran
- Scipy library: una collezione di algoritmi numerici e toolbox di dominio specifico per l'elaborazione dei segnali, ottimizzazione, statistica e molto altro. Della Scipy Library, si è utilizzato:

- ndimage: (n-dimensional image) contiene operazioni morfologiche ottimizzate con immagini binarie.

- MatplotLib: libreria di disegno utile per la realizzazione di grafici 2D e 3D
- Ipython: un'interfaccia su console che permette la veloce esecuzione di script Python, integrata con Spyder e il browser.
- Scikits: insieme di package con funzionalità più specifiche, come scikit-image e scikit-learn

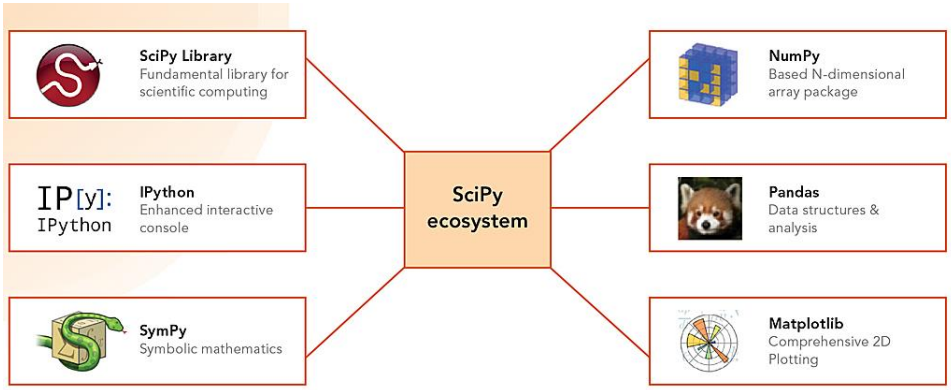

Figura 15.4 Struttura logica dell'ecosistema SciPy.

15.3.2 Scikit-Image

Figura 15.5 Scikit-Image Logo.

Scikit-image è una libreria per il processamento delle immagini che implementa algoritmi e utilities a scopo educativo, di ricerca o applicazioni industriali. È distribuita sotto licenza BSD (*open source license*) e fornisce della API ben documentate, vantando un team di sviluppo e una comunità di supporto internazionali [15].

Nell'implementazione degli algoritmi, si sono utilizzati i seguenti sotto package:

- morphology: per le operazioni morfologiche di base come erosione, dilatazione, apertura, chiusura, skeletonize, hit and miss, trasformata medial axis, rimozione di oggetti ecc.
- filters: per il metodo di Otsu e il filtro gaussiano

- draw: per il disegnare figure all'interno di un'immagine
- color: per effettuare conversioni tra spazi di colore
- measure: per l'algoritmo RANSAC e trasformata di Hough, ed effettuare il calcolo delle aree
- feature: per l'algoritmo di individuazione dei contorni con l'algoritmo di Canny
- io: per le operazioni di lettura e salvataggio delle immagini.

15.4 Struttura e Funzionamento

Nella realizzazione dei moduli, ho mantenuto il massimo della semplicità. A tale scopo Python, si è rivelato il linguaggio ideale. Nella stesura del codice infatti, ho mantenuto un approccio molto simile al paradigma funzionale, concentrandomi sulla definizione di funzioni. Questo approccio, ha portato notevoli vantaggi, come una ridotta quantità di codice necessaria, flessibilità e semplicità nel debugging e nel testing, possibilità riuso del codice. Data la natura sperimentale del progetto, la facilità di testing e debugging si è rivelata fondamentale. Di seguito è mostrato lo schema logico di relazione tra moduli

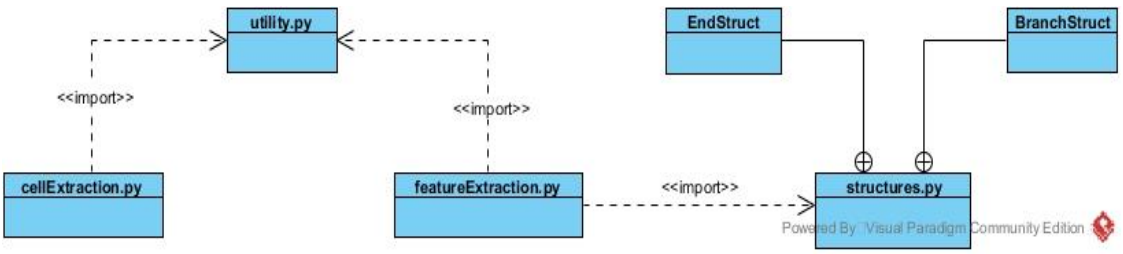

Figura 15.6 Relazione tra script.

Ogni modulo corrisponde ad un file con estensione ".py", cioè uno script, ed è possibile eseguirli sia su console, sia attraverso un controller che gestisce le chiamate di funzione. Come visibile dallo schema, vi è un modulo di utility, utilizzato per le operazioni di input/output, e fornisce servizi a livello di dati, file e cartelle. Questo modulo è condiviso in tutto il sistema, perciò è utilizzato anche nelle altre fasi. Il modulo *structures.py*, fornisce gli elementi strutturali per le operazioni di segmentazione dello scheletro. È l'unico modulo, in cui si è fatto uso di classi; infatti al suo interno sono implementate due *InnerClass*, che rappresentano le due tipologie di punti di uno scheletro branch-point e end-point. In realtà si è fatto uso di classi, al solo scopo di aumentarne la leggibilità

e la comprensibilità, quindi senza alcuno vincolo funzionale. I moduli *cellExtractor.py* e *featureExtractor.py*, forniscono gli algoritmi necessari per estrazione della cellula e estrazione delle features, con funzioni di supporto per le operazioni più complesse.

I moduli, così come gran parte del sistema, fa uso di programmazione parallela (multithreading). La programmazione parallela anche chiamata concorrenza, permette l'esecuzione di più processi in contemporanea, in modo da rendere più veloce l'esecuzione. Le moderne CPU, vantano *Core* multipli, che possono essere raddoppiati (virtualmente) attraverso tecniche di virtualizzazione. Il multithreading, sfrutta questa caratteristica, ottimizzando il lavoro della CPU.

Nei moduli, il multithreading è stato implementato in modo da eseguire operazioni di estrazione della cellula ed estrazione delle feature, su immagini multiple. Per l'implementazione si è fatto uso della libreria standard *multiprocessing*, che fornisce servizi di gestione ottimizzata dei *Thread*, attraverso i *ThreadPool.*

Negli algoritmi, sono stati adottati ulteriori meccanismi di ottimizzazione. In *cellExtractor.py* e nell'algoritmo di conta dei lobi vengono effettuati controlli su immagini monocromatiche e su maschere vuote, al fine di evitare l'inutile esecuzione degli interi algoritmi. Nell'estrazione delle features, invece vengono calcolate prima le features 1, 2, 5, 7 e se l'area del nucleo della cellula è maggiore di zero, vengono calcolate anche le altre, altrimenti, quelle mancanti avranno valore zero. Le features 3, 4 e 6, dipendono strettamente dalla presenza del nucleo, perciò se non è presente, non vengono calcolate, evitando inutili calcoli.

16 CONCLUSIONI

La realizzazione di questo progetto, mi ha permesso di confrontarmi con problematiche reali, relative all'informatica medica, nella realizzazione di un progetto coordinato. Ho avuto modo di partecipare allo sviluppo di un progetto, nato da una semplice idea ed evolutosi per diventare qualcosa di concreto e tangibile collaborando con aziende e specialisti sia del settore medico che informatico. È stata un'esperienza stimolante, che ha sicuramente contribuito alla mia crescita personale.

Ho avuto l'occasione di studiare e combinare diverse tecniche dell'image processing e applicarle in contesti differenti da quello informatico. La realizzazione di un dataset di discrete dimensioni, ha portato lo sviluppo del progetto su un livello più alto, rendendo i risultati e le problematiche più vicine a quelle reali e grazie allo sviluppo guidato da sperimentazione e test, e ai risultati ottenuti, ho una maggiore consapevolezza sulla reale affidabilità degli algoritmi proposti. Ho affrontato le maggiori problematiche nella realizzazione dell'algoritmo del conteggio dei lobi nucleari, dove non sono stati raggiunti totalmente i risultati sperati. Tuttavia ritengo che i test e le considerazioni proposte, siano utili al miglioramento degli algoritmi, che trattando problemi non deterministici, possono e devono essere continuamente migliorati. Le cellule in particolare, hanno una estrema variabilità nelle loro caratteristiche morfologiche, tale da renderne difficile la classificazione anche per uno specialista. Nel processo di testing è stata di fondamentale importanza, la presenza di un grande dataset a supporto della valutazione degli algoritmi. Naturalmente i test, descritti in questo lavoro, rappresentano solo una piccola ma significativa parte di quelli realmente realizzati, considerando che sono soprattutto i testi falliti, ad aver apportato maggior stimolo per la realizzazione di algoritmi migliori.

Dal lavoro è emerso che le features, sono sicuramente un elemento critico e di estrema rilevanza per il corretto funzionamento del sistema di riconoscimento. Gli algoritmi proposti in letteratura, hanno fornito ottime basi, ma non sono da considerarsi sufficienti a fornire una soluzione definitiva al problema. È necessario testare questi algoritmi su grandi dataset, in modo da poterne stimare realmente l'affidabilità e la precisione. Allo stato attuale l'estrazione della cellula leucocitaria dal ROI, risulta un algoritmo abbastanza maturo ma migliorabile, mentre l'algoritmo di conta dei lobi, ritengo sia più apprezzabile in un contesto qualitativo, piuttosto che quantitativo.

Il conteggio dei lobi anche se non totalmente esatto, fornisce comunque un valore significativo e pur non assumendo sempre il valore reale, fornisce un peso nella classificazione della cellula. L'errore nella "coerenza" con cui esso si presenta, stabilisce una relazione tra l'immagine e la tipologia di cellula che si vuole identificare ed è un elemento caratterizzante utile alla classificazione. Naturalmente tutti i risultati sono stati sottoposti al medico specialista ematologo, che li ha valutati positivamente, evidenziando anch'egli la rilevanza qualitativa della conta dei lobi e confermando l'analisi fatta sugli errori prodotti dagli algoritmi, queste valutazioni sono state confermate dai buoni risultati ottenuti nella successiva fase di classificazione.

Durante il lavoro ogni problema è stato profondamente analizzato per carpirne le cause e studiarne le possibili soluzioni, per poi testarle e valutarle. Lo sviluppo sperimentale è stato estremamente stimolante in quanto mi ha spinto a raffinare soluzioni che in altri contesti con dataset più piccoli, sarebbero state già soddisfacenti. In base a quest'ultima considerazione, posso affermare con certezza che questi algoritmi possono essere sicuramente migliorati, con dataset più grandi e diversificati. In ultima analisi posso ritenermi soddisfatto dei risultati ottenuti, considerando le difficoltà derivanti dalla classificazione di questo tipo di immagini, il lavoro svolto e le capacità acquisite.

17 SVILUPPI FUTURI

Come già sottolineato, ci sono significativi sviluppi futuri realizzabili in questo progetto.

In particolare, la tecnica di estrazione anche se soddisfacente, presenta comunque un buon margine di miglioramento. Una possibilità sarebbe quella di individuare i bordi della cellula e utilizzarli per correggere gli errori di estrazione.

Si potrebbero scartare cellule esplose, macchie di colore o altri corpi estranei, che vengono riconosciuti nella prima fase, effettuando un controllo sulla convessità del nucleo, in modo da filtrare i risultati prima dell'estrazione della cellula.

Il conteggio dei lobi è migliorabile affinando i parametri di taglio, o discriminando le cellule lobate da quelle non lobate in anticipo; a questo proposito si potrebbe continuare la sperimentazione sull'albero decisionale basato sull'analisi delle altre feature.

Ritengo necessario approfondire anche il numero e la tipologia di features da utilizzare, ed eventualmente sostituirle con di più rilevanti.

Anche la tecnica di scansione del vetrino è migliorabile, dovrebbe essere standardizzata, in modo da evitare variazioni di colore o messa a fuoco del vetrino.

Indice delle figure

Bibliografia

[1] A. Rosenfeld, J. L. Pfaltz, *Sequential Operations in Digital Picture Processing*, 1966.

[2] G. N. Srinivasan, G. Shobha, *Segmentation Techniques for Target Recognition,* Volume 7, Ottobre 2007.

[3] G. Qiao, G. Zong, M. Sun, J. Wang, *Automatic Neutrophil Nucleus Lobe Counting Based on Graph Representation of Region Skeleton*, 11/07/2012.

[4] L. Bianco, *Introduzione al fitting*, disponibile su: "www.diegm.uniud.it/fusiello/teaching/visione/progetti2002/bianco/node3.html", 12/12/2002.

[5] Leonova E. V. Chanturia A. V. Wismont, *Fisiologia patologica del sistema ematico*, 2009 disponibile su: "http://it.medicine-guidebook.com/patologicheskaya-fiziologiya_792_patologicheskie-form~1.html", 2014-2016.

[6] Motic, *Resources Specification sheets – MoticEasyScanPro,* disponibile su: "http://www.moticeasyscan.com/resources_specifications.html", [visto in data 14/09/2017].

[7] Nobuyuki Otsu, *A threshold selection method from gray-level histograms*, in *IEEE Trans. Sys., Man., Cyber.*, vol. 9, 1979, pp. 62–66.

[8] P. Carlucci, V. Bevilacqua, A. Guarini, *Progettazione di un sistema di supporto alle decisioni in ambito oncoematologico*, 2014.

[9] R. Fisher, S. Perkins, A. Walker and E. Wolfart, *Digital Filters*, disponibile su: "www.homepages.inf.ed.ac.uk/rbf/HIPR2/filtops.htm", 2003.

[10] R. Fisher, S. Perkins, A. Walker and E. Wolfart, *Gaussian Smoothing*, disponibile su: "www.homepages.inf.ed.ac.uk/rbf/HIPR2/gsmooth.htm", 2003.

[11] R. Fisher, S. Perkins, A. Walker and E. Wolfart, *Hit-and-Miss Transform*, disponibile su: "www.homepages.inf.ed.ac.uk/rbf/HIPR2/hitmiss.htm", 2003.

[12] R. Fisher, S. Perkins, A. Walker and E. Wolfart, *Skeletonization/Medial Axis Transform*, disponibile su: "www.homepages.inf.ed.ac.uk/rbf/HIPR2/skeleton.htm"2003.

[13] S. ADAMO, P. CARINICI, M. MOLINARO, G. SIRACUSA, M. STEFANINI, E. ZIPARO, *Istologia di V.Monesi, quinta edizione, ottava ristampa*, PICCIN, 2002, 2013.

[14] S. Eddins, Binary image *convex hull – algorithm notes*, disponibile su: "http://blogs.mathworks.com/steve/2011/10/04/binary-image-convex-hull-algorithm-notes", 2006.

[15] S. van der Walt, J. L. Schönberger, J- Nunez-Iglesias, F. Boulogne, J. D. Warner, N. Yager, E. Gouillart, T. Yu, *scikit-image: Image processing in Python*, 2014.

[16] SciPy Developers, *Numpy and Scipy Documentation*, disponibile su: "www.docs.scipy.org/doc/numpy/reference/generated/numpy.histogram.html", 2017.

[17] T.Y. Zhang, C.Y. Suen, *A Fast Parallel Algorithm for Thinning Digital Patterns*, Volume 7, Numero 3, Marzo 1984.

[18] V. Bevilacqua, D. Buongiorno, P. Carlucci, F. Giglio, G. Tattoli, *A supervised CAD to support telemedicine in hematology*, 2015

[19] Zanchettin, A. Cenedese, *Algoritmo di RANSAC: panoramica, confronti e applicazioni*, 2013.