Tesi

# bozza della tesi

## plugin in imagej

I componenti principali di ImageJ, che gli fanno anche da base, sono i Plugin che estendono e arricchiscono il Tool in modo semplice e efficace. Ogni elemento, anche l’interfaccia con cui accoglie l’utente, è creata attraverso i Plugin. Un Plugin si può sviluppare in Macro e Java ma in questo elaborato ci concentreremo sulle tipologie scritte con il secondo linguaggio di programmazione nominato. Qualsiasi utente può iniziare la stesura di un Plugin personalizzato. Attraverso il menu principale, scegliendo la voce “Plugins”, può iniziare la scrittura attraverso il comando new e scegliendo ilPlugin base prescelto. Il particolare ImageJ ne offre quattro:

1. Plugin
2. Plugin Filter
3. Plugin Frame
4. Plugin Tool

Per concentrarsi sulla individuazione e calcolo delle misure richieste per l’elaborato i plugin utilizzati sono della tipologia Plugin e Plugin Filter. In termini tecnici esse sono interfacce. PlugIn riporta un solo metodo chiamato run che prende come argomento un oggetto di tipologia String. Il metodo viene chiamato quando viene caricato il Plugin che si serve di questa interfaccia. L’argomento di classe String può essere anche vuoto, ovvero “”. Plugin Filter è leggermente più corposo. Al suo interno vi è il metodo setup. Quest’ultimo viene chiamato quando il Plugin viene caricato. Presenta due argomenti passati uno ti tipologia String che può essere vuoto e un oggetto di tipo ImagePlus che indica l’immagine attualmente attiva. Restituisce un valore di tipo int che indica se il metodo è andato a buon fine. Presenta anche un metodo run (medesimo nome della interfaccia citata sopra) ma con un obbiettivo diverso; utilizzano il metodo per elaborare l’immagine e presenta una serie di flag di controllo, pubbliche. Prende in ingresso un oggetto della classe ImageProcessor. Quest’ultima è volta ad applicare alcuni metodi base per manipolare immagini 2D.

## Plugin dell’elaborato

### introduzione

Nonostante ImageJ sia ricco di Plugin per poter trattare e calcolare molte features data in ingresso una immagine, nessun plugin prende tutte le misure necessarie all’obbiettivo proposto, o comunque non le raggruppa nello stesso Plugin. L’idea sarebbe poter sfruttare al meglio sui Plugin base su cui si basa ImageJ e estrarre le misure richieste relative ad immagini biomediche. In particolare, il Plugin che si avvicina di più a queste esigenze è “ParticlesAnalyzer”, avviabile dal menù “Analyze” con la voce “Analyze Particles…”. L’elemento più importante di ParticlesAnalyzer è la possibilità di individuare più oggetti in una immagine e calcolare le features su ognuno di essi. Questo avviene attraverso un Threshould manuale, definizione delle aree di interesse attraverso il plugin “Threshould” accessibile dal menu sotto le voci Image, Adjust e “Threshould”. Se si tratta di immagini in RGB, ovvero a colori, si usa invece “Color Threshould” accessibile da medesimo menù. Particles Analyzer implementa PlugInFilter e Measurements. Measuments è una interfaccia che lo collega ad un menù in cui si possono settare alcune misure ma ai fini dell’elaborato non sarà rilevante. Date le proprietà di questo Plugin la decisione presa per implementare e raggruppare le misure già esistenti è quella di creare un nuovo Plugin che estendesse Particles Analyzer e basarsi su di esso.

### Sviluppo e storia del plugin

Il Plugin prodotto è stato aggiornato volta per volta basandosi su tre stadi; il primo stadio prevedeva la implementazione di misure in B&W, sia già calcolate sia da calcolare; il secondo stadio prevedeva la implementazione delle misure in grey; il terzo stadio e ultimo prevedeva la implementazione delle misure in RGB. Durante la stesura del codice e laricerca di metodologie da applicare per inserire le formule, per la maggior parte note, all’interno del Plugin, si è voluto curare anche l’interfaccia in modo tale da poter scegliere se implementare tutte le misure proposte o solo alcune. Inoltre struttura del Plugin stesso è stato modificato più volte a causa, o grazie, a metodologie scoperte e applicate in corso d’opera che rendevano il codice comprensibile e logico. La particolarità dei Plugin di ImageJ è che si devono appoggiare completamente sulle classi o plugin già implementati su ImageJ stesso, una vera e propria libreria a parte volta solo ed esclusivamente all’utilizzo all’interno del tool.

### Struttura del Plugin

“Catch\_Parasite3” (??) è il nome attuale del Plugin elaborato. Esso implementa l’interfaccia PlugIn e al suo interno vi è un solo metodo chiamato run, e un vettore di ImagePlus di nome impVector. All’interno del metodo run viene avviato il plugin e se l’immagine è a colori viene trattata per elaborarla con i canali RGB e le nuove tre immagini derivanti dalla elaborazione vengono inserite all’interno del vettore impVector che verrà utilizzato per estrarle e calcolare le misure in RGB. Viene creato un oggetto appartenente inner class Parasite (??) che estende ParticlesAnalyzer ricavandone le proprietà necessarie per poter lavorare su più oggetti nell’immagine e altre funzioni per la visualizzazione di alcune misure implementate da ImageJ. Dopo aver creato l’oggetto Parasite, si avvia il suo setup. Se questo va a buon fine, ovvero il metodo setup rende una variabile intera uguale a “PlugInFiler.Done”, viene avviato anche il metodo run della stessa classe Parasite, dangogli come argomento l’ImageProcessor reso dall’oggetto ImagePlus attarverso il metodo getProcessor(). ImageProcessor è una classe astratta volta, come suggerisce il nome, a “processare” e trattare i dati dei pixel di un’immagine 2D. Infine viene richiamata dalla super classe Analyzer, il metodo volto a visualizzare i risultati ottenuti attraverso una ResultTable con titolo “Results”. Oltre a run vi è un metodo chiamato getImagePlusHSB volto al calcolo di alcune misure riguardanti l’ HSB stesso.

#### Parasite

Parasite è la inner class del Plugin principale, ovvero di Catch\_Parasite3. Viene creato un oggetto Parasite all’interno del metodo run di Catch\_Parasite3. La sua implementazione era necessaria perché estendendo la classe ParticlesAnalyzer non era banale far avviare il Plugin senza che avesse un supporto da parte di Catch\_Parasite3. Questo ultimo fa da tramite tra l’immagine e il plugin, inner classe. Parasite si occupa di estendere ParticlesAnalyzer e di implementare le nuove misure. Al suo interno vengono dichiarate una serie di variabili booleane necessarie per selezionare solo le misure scelte dall’utente e in particolare un intero “measure” che accoglierà i valori interi corrispondenti alle misure già implementate da ImageJ. Ogni valore intero ha un corrispondente specifico come misura. Seguono i metodi necessari per aggiungere le misure implementate nel plugin; molti di essi sono Override dei metodi della super classe. Il primo metodo importante è showDialog che non prende argomenti in ingresso e restituisce un valore di tipo intero. Esso è un Override e richiama il metodo della super.classe. Questo visualizzerà un menu iniziale già implementato da Particles Analyzer. Questo metodo si occupa anche di verificare se il setup è andato a buon fine. Verificando nella struttura della super classe estesa si può denotare che showDialog è chiamato all’interno del setup stesso. L’Override era necessario per aggiungere un nuovo menù per settare le misure da visualizzare. Inoltre verifica se l’immagine è a colori o meno. Questo passo è importante per visualizzare le misure per RGB oppure per non visualizzarle nel menu per la scelta delle features. All’interno del metodo viene richiamato genericDialog (??) che comporta la visualizzazione di un nuovo menu attraverso una nuova GenericDialog. Vengono visualizzate le scelte attraverso checkBox e raggruppate a seconda della implementazione o della tipologia di features da presentare. In ordine abbiamo: le features implementate da ImagJ, le features implementate durate lo studio e la creazione dell’elaborato riguardanti le misure per immagini in bianco e nero, le features riguardanti le immagini in grigio, le features riguardanti le immagini a colori. Queste ultime non appariranno se l’immagine non è a colori. Allo stesso modo le misure “min and max” per le immagini in grey non apparirà se l’immagine è a colori poiché darebbe valore zero. I valori selezionati vengono immessi nel vettori booleani di appoggio riguardanti le misure di ImageJ, le misue in Bianco e Nero, misure in grigio e misure in RGB. Il metodo setMeasureExtended() si occuperà di recuperare i valori booleani dei vettori; per le misure di ImageJ dipendentemente dal valore positivo o negativo verranno aggiunte le costanti alla variabile intera measure, che verrà poi passata a Analyzer.setMeasurements(measure). Per le misure implementate dall’elaborato verranno presi i valori dipendentemente dall’indice e immessi nelle variabili booleani i quali indicano attraverso il loro nome quale misura verrà calcolata a seconda del suo valore vero o falso; per esempio “doConvexArea” se verrà settato come vero nella ResultTable apparirà tale valore. L’Override più importante è del metodo saveResults; la super classe si occupa di creare l’oggetto Roi che prenderà il nome roi nel metodo, l’area di interesse che occupa l’oggetto singolo analizzato nella immagine e il suo ImageStatistics ovvero un oggetto che si occupa del calcolo delle misure di natura statistica dell’oggetto. Dopo aver richiamato il metodo della super classe, vengono dichiarate alcune variabili necessarie per il calcolo di alcune misure; un oggetto della classe Polygon reso dall’oggetto roi attraverso il metodo getConvexHull(); in questo caso ImageJ usa una classe di Java Polygon per trattare con il convexHull della regione di interesse. Grazie alla creazione di tale oggetto, ne viene calcolata area e perimetro del convex hull stesso, chiamato un metodo della classe Parasite, getArea e getPerimeter. Per questione di comodità viene dichiarata e settata la variabile fert, un vettore contenente i dati relativi al feret attraverso un metodo della roi e il perimetro sempre attraverso un metodo. Ultimo ma non meno importante viene anche dichiarato un vettore hist e popolato dall’istogramma della roi, attraverso la restituzione da stats, il suo ImageStatistics collegato. Con una serie di if consecutivi, viene verificata la necessità di calcolare le features attraverso i booleani settati precedentemente. Ognuna di esse ha una formula a sé spesso applicata nella stampa. Solo le features Haralick Ratio, Bending Energy, Entropy, IntensitySum, Uniformity, Smothness si è preferito farle a parte in un metodo per avere un codice più pulito e comprensibile. Se l’immagine è in RGB si controllano anche le misure relative a quella tipologia. Dunque si crea un oggetto ImageStatistics dalle immagini ricavate inizialmente nel plugin, 0 red 1 green e 2 blue, e si setta la roi ad esse. Stesso ragionamento si può applicare alla seconda parte delle misure a colori ovvero le riguardanti al HSB, hue, saturation e brightness. La differenza tra le due tipologie di misure è che per la prima vi era una funzione offerta dall’oggetto ChanelSplitter che crea un vettore di ImagePlus per i tre canali mentre per le misure in HSB non vi era una funzione simile dunque è stata implementata una funzione a parte, chiamata getImagePlusHSB all’interno del Plugin stesso.

Dopo l’Override di saveResults seguono i metodi di appoggio per calcolare l’area e il perimetro del convexHull, Bending Energy, HeralickRatio, Entropy, InsentisitySum, Uniformity e Smothness. A seguire una serie di metodi di appoggio per la distanza, calcolo del logaritmo in base due e calcolo di operazioni per i vettori.

### Misure INserite calcolte da imagej

|  |  |
| --- | --- |
| Nome della misura | Descrizione |
| Area | Area della regione di interesse |
| Perimeter | Perimetro della regione di interesse |
| Centroid | Centroide o centro geometrico della regione, ovvero la posizione media di tutti i suoi punti inteso anche come media aritmetica della posizione di ciascuno di essi. |
| Center of mass | Centro della massa della regione, punto geometrico corrispondente al valore medio della distribuzione della massa del sistema nello spazio. |
| Bounding Rectangle | Valore dell’insieme dei punti del rettangolo della scatola di delimitazione (bounding box) della regione |
| Fit Ellipse |  |
| Shape descriptors |  |
| Feret’s diameter |  |
| Integrated density |  |
| Area\_fraction |  |
| Stack Position | Posizione della immagine nello stack di immagini |
| Mean | Media del valore grigio presente nella regione di interesse |
| Skewness | Grado di assimmetria della distribuzione dei valori di intesità dei livello di grigio |
| Kurtosis K | La curtosi è un allontanamento della normalità distributiva, rispetto alla quale si verifica un maggiore appiattimento o un maggiore allungamento, applicata ai valori di grigio  Una misura del grado in cui la distribuzione dei valori di intensità di grigio si discosta dalla curva gaussiana |
| Median | La mediana dei valori dei pixels nell’area di interesse. |
| Min and Max | Il minimo e il massimo della intensità di grigio presente nella regione, nelle immagini a colori non è possibile calcolarlo. |

### Misure implementate BW

|  |  |
| --- | --- |
| Nome della misura | Descrizione |
| Convex Area | Area del poligono convesso.  Somma di tutte le distanze tra i; dato un punto A e un Punto A-1 e divisione per due. (Tratta il convex hull come se fosse composto da tanti triangolo?) |
| Convex Perimeter | Perimetro del poligono convesso  Somma delle distanze tra tutti i punti del contorno del Convex Hull |
| MinR (?) | Raggio del cerchio inscritto con punto di origine uguale al centro della massa interessata.  Divisione del minimo diametro per due |
| MaxR (?) | Raggio del cerchio inscritto con punto di origine uguale al centro della massa interessata  Divisione del massimo diametro, chiamato feret, per due |
| AspRatio | Chiamato anche Feret Ratio o Eccentricy o Rectangular Ratio |
| Roundness |  |
| ArEquivD | Diametro del cerchio con l’equivalente area. |
| PerEquivD | Diametro del cerchio con l’equivalente perimetro. |
| EquivEllAr | Area dell’ellissi con il Feret come asse maggiore e il Breath asse minore. |
| Compactness |  |
| Concavity |  |
| Convexity | Chiamata anche rugosity o roughness |
| Shape | Chiamata anche Thinness ratio. |
| RFactor |  |
| ArBBox |  |
| Rectangularity |  |
| Modification ratio | ( |
| Sphericity | Radius ratio.  MinR/MaxR |
| Elongation | L’inverso di circularity. |
| Haralick Ratio | Calcolo della media dei raggi e della deviazione standard di essi, tramite metodo getHaralickRatio. |
| Bending Energy (?) |  |

### Misure implementate grey e a colori

|  |  |
| --- | --- |
| Nome della misura | Descrizione |
| Entropy | Misura della casualità, tramite metodo getEntropy.  Data l’area e l’istogramma dei grigi, somma della probabilità che avvenga quel livello di grigio e lo stesso valore in logaritmo di due, per tutti i valori dell’istogramma, dunque fino a 256. Scarto del livello di grigio nel caso non ci siano pixel con tale gradazione. |
| Intensity | Somma delle intensità di livello di grigio presenti nell’oggetto, tramite metodo getIntensitySum.  Dato l’istogramma dei grigi, somma dei livello per la sua quantità di pixel aventi quel livello e somma totale. |
| Square Intensity Sum | Radice quadrata di Intensity. |
| Uniformity | Una misura sulla uniformità, tramite metodo getUniformity.  Dato l’istogramma dei grigi e l’area calcolo della probabilità che il livello di grigio considerato si presenti nell’area. Tale probabilità viene elevata al quadrato. Vengono sommati tutti i risultati. |
| Smoothness R | Una misura sulla relativa levigatezza nella regione. Se R è uguale a 0 allora vi sono valori uguali in tutta la regione se invece si avvicina ad 1 oppure è uguale ad 1 la regione ha grandi escursioni di valori. Tramite metodo getSmoothness.  Data la varianza si calcola la varianza normalizzata e si esegue la seguente formula: |

Le misure per immagini a colori sono state fatte sfruttando la proprietà per cui una immagine in grigio è uguale alla somma dei valori in rosso, in verde e in blu diviso tre oppure:

Di conseguenza si può trattare l’immagine e estrapolarne la sua versione grigia ma con avente solo il rosso, il verde e il blu. Alle immagini, i cui valori sono contenute in un vettore ImagePlus da tre locazioni e non vengono visualizzate poi, si applicano le stesse formule delle immagini in grey. Di conseguenza la media e la deviazione standard dei valori RGB, presi separatamente sono stati estrapolati attraverso l’oggetto ImageStatistics e applicando alle nuove ImagePlus ottenute lo scheletro delle immagini di interesse con il metodo setRoi(roi).

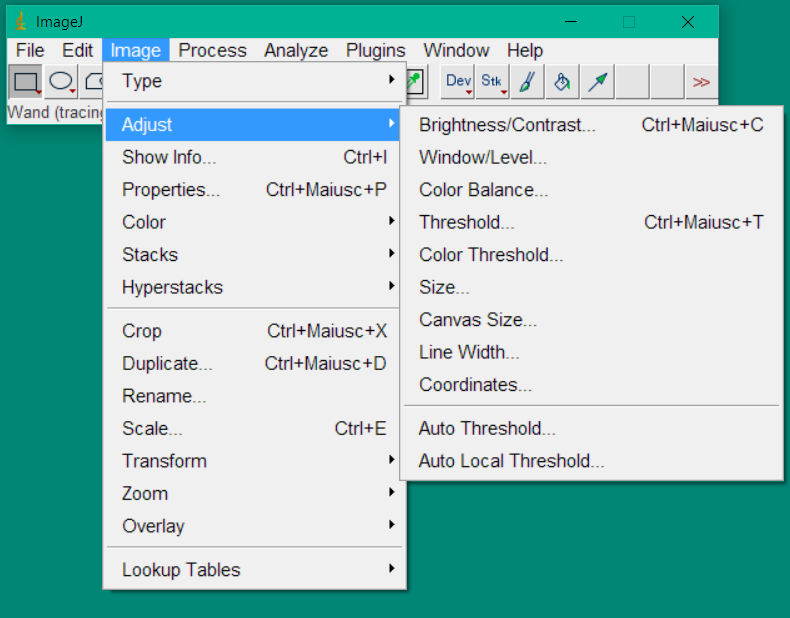
|  |  |
| --- | --- |
| Average red color, R mean | Media dei valori in rosso |
| Average green color, G mean | Media dei valori in verde |
| Average blue color, B mean | Media dei valori in blu |
| Red color std deviation, R std | Deviazione standard dei valori in rosso |
| Green color std deviation, G std | Deviazione standard dei valori in verde |
| Blue color std deviation, B std | Deviazione standard dei valori in blu |
| Square root of the value mean of channel R, G, B | Radice quadra della media del rosso, del verde e del blu. |
| Avarage RGB | La media RGB |

Sono state calcolate misure riguardanti HSB ovvero tonalità, saturazione e luminosità; esso è un metodo additivo di composizione dei colori. Dai tre parametri, la tonalità varia partendo convenzionalmente dal rosso primario a 0°, passando per il verde primario a 120° e il blu primario a 240° e quindi tornando al rosso a 360°; per saturazione si intende l’intensità e la purezza della singola tonalità mentre la luminosità e un’indicazione della sua brillantezza. Prendendo singolarmente tali fattori, si ottengono immagini in grey differenti dalla originale o dalle corrispondenti nei canali RGB. Dunque è possibile calcolare la misura della media e della deviazione standard. Per ricavare tale immagini il Plugin si serve di un metodo dedicato. Quest’ultimo ha il nome di getImageStackHSB() che prende come parametro l’ImagePlus, variabile globale. Al suo interno vengono creati gli ImageStack corrispondenti a Hue, Saturation e Brightness. L’immagine sotto forma di oggetto ImagePlus viene salvata in un quarto ImageStack. Da esso si ricava un oggetto cp di classe ColorProcessor. Quest’ultimo è una sotto classe di ImageProcessor, il quale contiene i dati dei pixel. L’oggetto cp, assegna a tre vettori di tipologia byte i valori dei pixel di HSB. Gli ImageStack vengono settati attraverso i vettori che contengono i pixel. Successivamente, dagli ImageStack si ricavano le ImagePlus immesse in un vettore ImagePlus e restituito al metodo run del Plugin esterno. In saveResults verrà usato il vettore per ricavarne le misure.

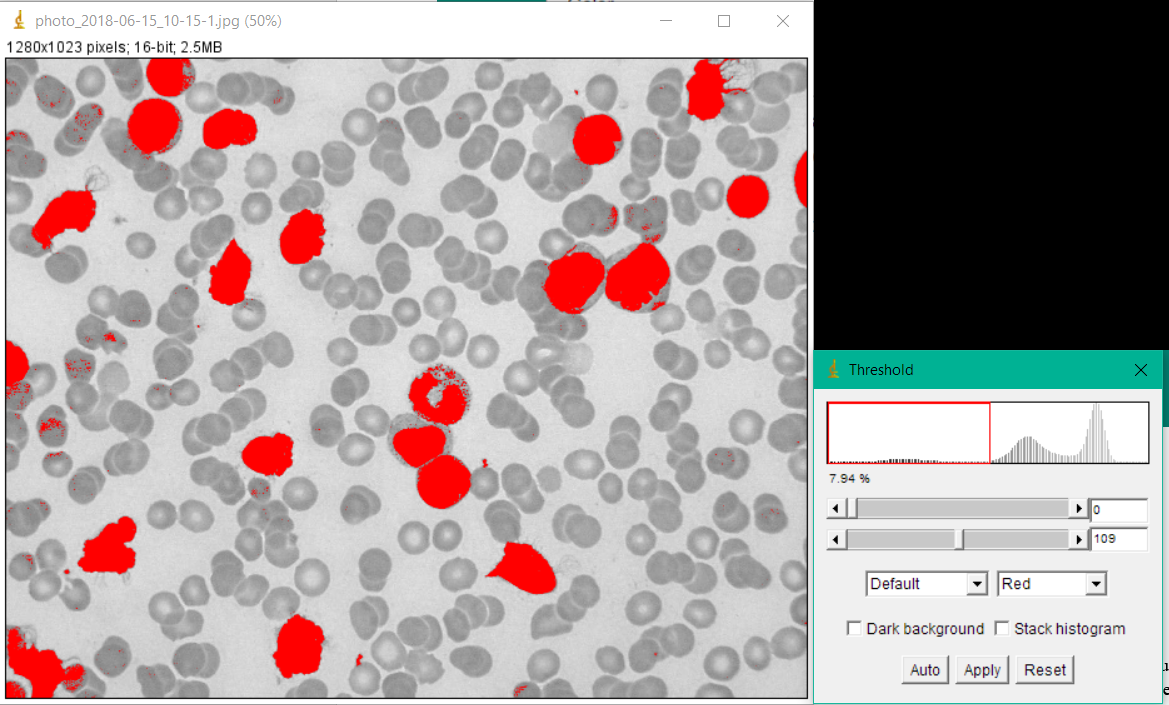
|  |  |
| --- | --- |
| Average Hue color, H mean | Media dei valori in Hue |
| Average Saturation color, S mean | Media dei valori in Saturation |
| Average Brightness color, B mean | Media dei valori in Brightness |
| Hue color std deviation | Deviazione standard dei valori in Hue |
| Saturation color std deviation | Deviazione standard dei valori in Saturation |
| Intensity color std deviation | Deviazione standard dei valori in Brightness |

### come usare il plugin

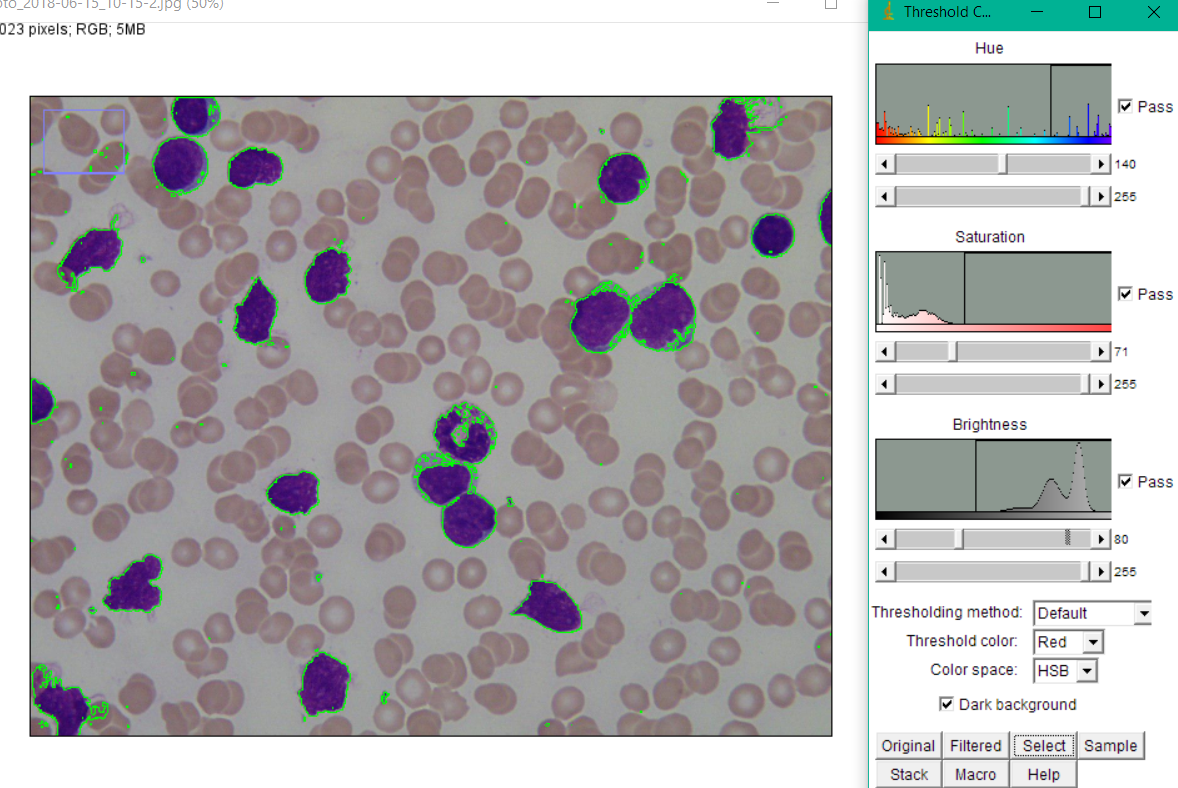
Il Plugin dell’elaborato ha come scopo semplificare il più possibile il calcolo immediato delle misure riguardanti immagini biomediche. Per poter iniziare, bisogna settare l’immagine manualmente e individuarne le soglie. Nel caso di immagini bianche e nere e in grigio si procede accedendo al comando di ImageJ Threshold. Si accede attraverso la combinazione di tasti Ctrl+Maiusc+T oppure dal menu stesso, selezionando Image>>Adjust e Threshold come illusa la figura.



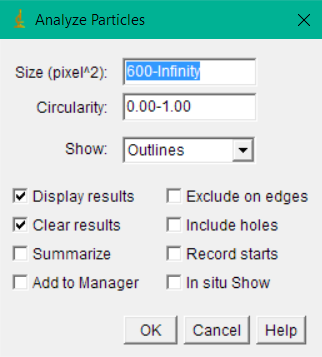
Per poter definire le soglie, si spostano i cursori e si settano a proprio piacimento le soglie e dunque le regioni di interesse come nell’esempio. Una volta decisi si esce tramite comando “X”. “Apply” rende l’immagine bianco e nera.



Per quanto riguarda la definizione delle soglie in una immagine a colori si sceglie dallo stesso menù “Color Threshold”. Qui è possibile settare attraverso i valori di Hue, Saturation e Brightness, sempre spostando i cursori e poter scegliere altre opzioni a proprio piacimento. Terminato il settaggio si seleziona il comando “Select” e l’immagine si presenta come in figura.



Una volta fatte queste operazioni preliminari, è possibile avviare il plugin dal menu Plugins>>Catch\_Pariste3. Il primo menu che si presenta è possibile scegliere la grandezza minima dell’oggetto analizzato e a seguire quella massima e la sua circolarità. Attraverso il menu a tendina si può scegliere il trattamento della immagine finale. Le scelte sono: nothing, per cui l’immagine non viene trattata, outlines per cui l’immagine viene trattata e come risultato si ha una seconda immagina bianca dove si vedono solo ed esclusivamente i contorni degli oggetti con una etichetta che sta ad indicare il numero dell’oggetto, bare outlines per cui si vedono solo i contorni degli oggetti, ellipses per cui l’immagine risultante è a sfondo bianco con delle ellipsi orientate al posto degli oggetti individuati, mask che restituisce l’immagine della maschera in bianco e nero, nero per gli oggetti e bianco lo sfondo, count mask (??), overlay che riprendendo l’immagine originale etichetta gli oggetti e infine overlay masks che riprende l’immagine originale e nelle zone dove si trova la maschera individuata sostituisce con un colore celeste chiaro. Le altre opzioni che si possono settare attraverso checkbox sono display results per visualizzare appunti i risultati, clear results per cancellare i risultati precedenti e summarize, Add to Manager, Exclude on Edges, Include holes, records starts, in situ Show.



Selezionato “Ok” si apre un secondo menù in cui si possono selezionare le misure. Esse sono separate da due fattori; le prime sono le misure presenti di default da ImageJ e dal Plugin Analyzer Particles che viene esteso dalla inner class del Plugin dell’elaborato. Dopo di che, seguono le misure implementate e aggiunte suddivise per tipologia di immagine e misura riguardante essa.

