

ALMA MATER STUDIORUM
UNIVERSITÀ DEGLI STUDI DI BOLOGNA

Seconda Facoltà di Ingegneria - Sede di Cesena
Corso di Laurea in Ingegneria Informatica

INTERFACCIA GRAFICA PER
L'ELABORAZIONE DI IMMAGINI
MICROSCOPICHE DI COLTURE BATTERICHE
GENETICAMENTE MODIFICATE

Elaborata nel corso di: Elaborazione delle Immagini LM

Tesi di Laurea di:
CRISTIAN ARCAROLI

Relatore:
Prof. ALESSANDRO
BEVILACQUA

Correlatori:
Dott.Ing. ALESSANDRO
GHERARDI
Prof. EMANUELE
DOMENICO GIORDANO

ANNO ACCADEMICO 2010-2011
SESSIONE III

PAROLE CHIAVE

Image Analysis

Microscopy

Automated system

Bioengineering

Synthetic biology

Il successo è individuale.

Indice

Introduzione	V
1 Dominio di applicazione	1
1.1 Fisico	1
1.1.1 Fluorescenza	1
1.2 Biologico	2
1.2.1 La biologia sintetica	2
1.2.2 Green Fluorescent Protein	3
1.2.3 Misure quantitative	4
1.3 Strumentazione e problemi annessi	5
1.3.1 Microscopio	5
1.3.2 Fotocamera	7
1.4 Informatico	13
1.4.1 Colori	13
1.4.2 Formato dell'immagine	16
2 Tecniche di elaborazione delle immagini	17
2.1 Tipologie di immagini	17
2.2 Convoluzione	18
2.3 Filtraggio	19
2.4 Segmentazione	20
2.4.1 Sogliatura	21
2.5 Edge detection	24
2.5.1 Sobel operator	24
2.5.2 Canny Operator	25
2.6 Operatori morfologici	26
2.6.1 Hit and Miss	27

2.6.2	Opening	27
2.6.3	Erosione	27
2.6.4	Dilatazione	28
2.7	Labeling	28
2.8	Minimum Enclosing Rectangle	29
2.9	Funzione di risposta	29
3	Progetto	33
3.1	Motore di elaborazione	33
3.1.1	Analisi dei dati di input	34
3.1.2	Scelta del metodo di segmentazione	36
3.1.3	Misurazioni	40
3.1.4	Parameter Tuning	40
3.1.5	Macchina a stati	48
3.2	Interfaccia grafica	50
3.2.1	Layout	50
3.2.2	Interazione con l'utente	52
3.2.3	Interazione dell'utente	52
3.2.4	Grafici	55
3.2.5	Stile	56
4	Implementazione	57
4.1	Struttura della soluzione	58
4.1.1	MainApplication	58
4.1.2	ImagingChart	58
4.1.3	Measurement	60
4.2	Interazione delle parti	65
4.3	Dettagli implementativi	66
4.3.1	Sauvola con integral images	66
4.3.2	Aggiornamento interfaccia	68
4.3.3	Particolari dei grafici	68
4.3.4	Implementazione Algoritmi	68
5	Risultati	71
5.1	Risultati	71
5.2	Presentazione del prodotto finito	74
6	Conclusioni e possibili sviluppi	79

A Sviluppo e calcolo della funzione di risposta	81
A.1 Sviluppo	81
A.2 Implementazione	82
Ringraziamenti	87

Introduzione

Nel 1981 la Xerox vendeva il suo primo computer con una *Graphical User Interface* a disposizione dell'utente per compiere le azioni più comuni. Da quel momento le GUI hanno iniziato la loro ascesa ed ora sono fondamentali per la maggior parte dei software prodotti. La *GUI* (al contrario della *CLI Command Line Interface*) permette all'utente di lavorare manipolando oggetti grafici invece che scrivendo testo, consentendo un più alto livello di accessibilità per ogni tipo di utente. Oramai le GUI sono talmente diffuse che spesso nell'acronimo viene omessa la lettera G, poiché si considera scontato che l'interfaccia utente sia di tipo grafico. Le GUI più comunemente utilizzate seguono un paradigma *WIMP* (Windows, Icons, Menus, Pointer) denotando uno stile di interazione che utilizza questi semplici elementi fondamentali. Recentemente, tuttavia, si parla di paradigma *post-WIMP* poiché i metodi di interazione e gli spazi a disposizione si stanno rapidamente evolvendo, anche se questo fenomeno è più evidente nel campo dei dispositivi mobili.

Grazie all'avvento di sensori ottici a stato solido (e.g. CCD, CMOS), sempre più complessi e con risoluzione sempre più elevata, c'è stato un notevole progresso nella scienza e nella ricerca, e questi sensori sono stati impiegati in svariati differenti campi applicativi. Denominatore comune di questi campi è la volontà di sostituire, o quantomeno affiancare, l'analisi umana, nel migliore dei casi soggettiva e non ripetibile, nel peggiore diretta, spesso grossolana o distratta. L'uomo, infatti, offre solo un'interpretazione soggettiva dei risultati poiché il suo "sistema di acquisizione ed elaborazione" (cioè, occhi e cervello) cambia da individuo ad individuo. L'uomo inoltre è molto più abile ed efficace nell'attività di riconoscimento, ma sicuramente un elaboratore è più efficace, ed efficiente, nelle attività automatiche e ripetitive. I sensori

sono molto utili se affiancati ad un’elaborazione digitale poiché possono fornire all’utente finale misurazioni accurate, altrimenti di difficile reperimento. Un esempio di queste misurazioni sono quelle *radiometriche* oppure *morfometriche*.

Uno dei settori in cui queste tecnologie sono impiegate è la medicina/biologia e in particolare in questo lavoro si fa riferimento alla biologia sintetica.

La *biologia sintetica* è quella branca della biologia che si occupa di studiare organismi geneticamente modificati utilizzando *black-box* di materiale biologico (realizzati e resi disponibili da terzi) dei quali si conosce solo la funzione ingresso-uscita finale. Attraverso il concatenamento di questi blocchi con specifiche funzionalità, si cerca di indurre l’organismo ospitante a svolgere alcune determinate funzioni, quali ad esempio la produzione di antibiotici contro particolari patogeni. Per poter predire il comportamento dell’organismo geneticamente modificato, non è necessario conoscere le caratteristiche specifiche di ogni blocco utilizzato, ma partendo dalla funzione di risposta il lavoro cruciale risulta essere il dimensionamento e la regolazione dei parametri dei vari componenti utilizzati al fine di creare organismi “operai” stabili e indotti ad essere positivamente produttori. A tal fine, una precisa e accurata misura delle caratteristiche dei sistemi biologici in gioco risulta fondamentale per la realizzazione del sistema finale. In particolare, per studiare e analizzare il comportamento dei componenti principali che hanno un ruolo attivo nella trascrizione (e.g. promotori, plasmidi, terminatori) si effettua un’operazione di *labeling* utilizzando dei geni reporter tipicamente fluorescenti in grado di provare il corretto funzionamento di una catena di componenti. Uno dei geni reporter più comunemente inserito in batteri sintetici per verificare il corretto funzionamento di una catena di blocchi è il gene che codifica la proteina *Green Fluorescent Protein* (GFP), in grado di emettere luce nel campo del verde se irradiata con raggi di lunghezza d’onda nel campo del blu. La misura quantitativa del segnale emesso da queste proteine automaticamente prodotte nel caso un processo sia avvenuto con successo, fornisce indicazioni sull’efficienza della catena dei componenti sintetici aggiunti all’organismo geneticamente modificato.

La motivazione che ha portato ad affrontare il presente lavoro di tesi è l’assenza di uno strumento che permetta ad un biologo di analizzare

efficacemente e dettagliatamente immagini di una colonia di batteri geneticamente modificati. Lo strumento tipicamente utilizzato fino ad oggi è lo *spettrofluorimetro*, ovvero uno strumento che fornisce un'analisi dello spettro di emissione del campione analizzato. Questo strumento presenta varie limitazioni, tra cui la possibilità di ottenere esclusivamente una misura globale del campione.

Nel presente lavoro di Tesi ci si propone di progettare e realizzare uno strumento di misura che risolva le limitazioni poste dallo spettrofluorimetro. Il software sarà corredato di una GUI che assottigli la dicotomia sovente presente tra il progettista di un software e il suo utilizzatore, permettendo a quest'ultimo di muoversi in un dominio conosciuto senza dover aver forti conoscenze informatiche a priori. In particolare, l'interfaccia creata ha lo scopo principale di affiancare e guidare in maniera estremamente *user's friendly* i biologi durante l'analisi di immagini di batteri geneticamente modificati con tecniche di biologia sintetica. Scopo dell'interfaccia grafica creata è appunto quantificare il segnale fluorescente emesso dai batteri, aiutando l'utente a risolvere problemi tipici delle immagini acquisite in microscopia quali risoluzione della funzione di risposta e vignetting.

Per testare lo strumento realizzato sono state utilizzate immagini di colture di batteri, acquisite in campo chiaro e contrasto di fase utilizzando un comune microscopio ottico accoppiato ad una telecamera digitale. In particolare, per l'acquisizione delle immagini è stato utilizzato il software MyAct3 realizzato in linguaggio C# dall'Ing. Alessandro Gherardi del gruppo di ricerca Computer Vision Group (CVG) dell'Università di Bologna, coordinato dal Prof. Alessandro Bevilacqua. MyAct3 è attualmente rilasciato gratuitamente in versione Beta ed è stato sviluppato nel contesto di un progetto di collaborazione con il modulo di Rigenerazione Tessutale Ossea dell'Istituto Ortopedico Rizzoli di Bologna diretto dal Dott. Enrico Lucarelli. Attualmente il software è stato concesso in utilizzo gratuito al Laboratorio di Ingegneria Cellulare e Molecolare (ICM) della Seconda Facoltà di Ingegneria di Cesena, coordinato dal Prof. Emanuele Domenico Giordano, e al Laboratorio di Bioscienze dell'Istituto Scientifico Romagnolo per lo Studio e la Cura dei Tumori (IRST) di Meldola, coordinato dal Dott. Wainer Zoli. L'interfaccia grafica sviluppata nel presente lavoro di Tesi sarà in seguito inclusa nella nuova versione del software MyAct3

per aumentarne le funzionalità e per fornire in output le misurazioni richieste.

Lo scopo del presente lavoro di Tesi è quindi fornire al biologo uno strumento estremamente *user's friendly* per la quantificazione di segnali fluorescenti partendo da immagini acquisite in fluorescenza con un normale microscopio *wide-field*.

La tesi presenta un primo capitolo in cui viene analizzato più approfonditamente il problema in tutti i suoi aspetti, introducendo anche la descrizione della strumentazione utilizzata. Nel secondo capitolo vengono spiegate le tecniche di elaborazione delle immagini utilizzate per risolvere il problema. Il terzo capitolo descrive il progetto dell'interfaccia grafica, mentre il quarto capitolo analizza l'implementazione del progetto. Il quinto capitolo discute i risultati ottenuti con l'utilizzo della GUI per l'applicazione delle tecniche di elaborazione dell'immagine e un sesto capitolo trae le conclusioni.

Capitolo 1

Dominio di applicazione

In questo capitolo vengono presentati i domini di applicazione in cui ci si muoverà nella tesi, l'approfondimento di questi potrà ad una corretta analisi dei requisiti per il software da sviluppare.

1.1 Fisico

1.1.1 Fluorescenza

La fluorescenza è una forma di luminescenza ed è quel fenomeno in cui una sostanza emette una radiazione elettromagnetica nel campo del visibile dopo aver assorbito un fascio luminoso o un altro tipo di radiazione elettromagnetica. Nella maggior parte dei casi la luce emessa ha una lunghezza d'onda maggiore, e quindi una minore energia della radiazione assorbita.

Quando la radiazione elettromagnetica assorbita è intensa, per un elettrone è possibile assorbire due fotoni e questo tipo di assorbimento conduce all'emissione di una radiazione con lunghezza d'onda minore rispetto quella assorbita.

La radiazione emessa potrebbe anche essere della stessa lunghezza della radiazione assorbita, in questo caso si parla di "fluorescenza di risonanza" [16]. L'esempio più appariscente di fluorescenza si ha quando la radiazione assorbita è nella banda ultravioletta dello spettro (quindi invisibile all'occhio umano) e la luce emessa è nella banda visibile. Fisicamente l'emissione di luce è dovuta ad un salto di stato nella

configurazione elettronica nella molecola colpita, per la precisione un elettrone viene portato ad un livello di energia superiore e durante il successivo decadimento verso lo stato originario viene emessa l'energia accumulata sotto forma di radiazione luminosa.

Le molecole in grado di originare transizioni elettroniche risultanti nel fenomeno della fluorescenza sono note come marcatori o fluorocromi. I fluorocromi che vengono legati a molecole più grandi (acidi nucleici, lipidi, enzimi o proteine) attraverso assorbimento o legame covalente sono chiamati anche fluorofori. Questi ultimi possono essere divisi in due grandi classi: intrinseci (e.g. GFP, amminoacidi aromatici, neurotrasmettitori, porfirine) e estrinseci (e.g. coloranti sintetici, molecole biochimiche modificate).

La fluorescenza è spesso usata nella scienza come metodo non distruttivo per tener traccia o analizzare molecole biologiche attraverso il monitoraggio della loro emissione fluorescente.

1.2 Biologico

Per quanto riguarda le conoscenze biologiche è stato necessario confrontarsi con esperti in materia che in questo caso fanno anche la parte dei committenti e quindi degli utilizzatori finali. Le informazioni riguardanti questo ambito, descritte di seguito, sono state in gran parte prese dal precedente lavoro in [13].

1.2.1 La biologia sintetica

La biologia sintetica è un'area di ricerca che combina biotecnologia e ingegneria, questa disciplina utilizza semplici parti di sistemi biologici naturali per formare sistemi biologici più complessi. L'obiettivo principale è quindi formare dei sistemi biologici complessi non ancora presenti in natura.

Per raggiungere questo obiettivo viene utilizzato il DNA come se fosse la memoria fisica su cui porre le istruzioni del software e l'apparato di sintesi proteica è paragonabile all'hardware di un calcolatore. Il DNA è sempre stato difficile da trattare fin dagli anni '70, ma oggi è possibile isolare una regione specifica di un genoma, produrne un

numero illimitato di copie e determinare la sequenza dei suoi nucleotidi in poco tempo. Alcune tecniche di ingegneria genetica permettono di modificare e trasferire il gene isolato in una linea germinale di un essere vivente in modo da diventarne una nuova parte funzionale ereditabile, conferendogli così nuove caratteristiche.

Questa procedura è possibile tramite l'utilizzo di DNA ricombinante, un DNA non esistente in natura ma creato combinando materiale genetico da origini differenti. Il DNA creato viene poi inserito all'interno di altri organismi con dei *vettori* biologici (e.g. molecole circolari, plasmidi, o strutture derivate da virus). Per un maggiore approfondimento sul DNA ricombinante rimando alla tesi in [13].

1.2.2 Green Fluorescent Protein

La green fluorescent protein (GFP) fu scoperta nel 1962 da Shimomura et al. [15] come una proteina associata all'equorina, la celebre proteina chemioluminescente ricavata dalla medusa *Aequorea Victoria*. Studi successivi dello stesso gruppo sullo spettro di emissione della GFP (che si affievolisce verso i 508 nm) mostrarono che la bioluminescenza verde dei tessuti viventi della *Aequorea* tendeva a spegnersi verso tale lunghezza d'onda, mentre la chemioluminescenza della pura equorina, che è blu, tendeva a spegnersi vicino a 470 nm, vicino cioè ad uno dei picchi di eccitazione della GFP. Da questa osservazione conclusero che la GFP convertiva l'emissione blu dell'equorina nella luminescenza verde delle cellule intatte e degli animali. Morin e Hastings, nel 1971 [8], trovarono la stessa sfasatura di colore nel celenterato di *Obelia*, un idroide, e nella *Renilla*. In seguito la GFP venne purificata e cristallizzata, venne misurato lo spettro di assorbimento e la quantità di fluorescenza prodotta. Il salto che ha determinato il successo di questa molecola avvenne con il clonaggio del gene da parte di Prasher et al. (1992) [10] e la dimostrazione che l'espressione del gene crea fluorescenza anche in altri organismi; perciò il gene contiene tutte le informazioni necessarie per la sintesi del cromoforo, non essendo necessari enzimi specifici della medusa. Nel 2008 è stato assegnato a Osamu Shimomura, Martin Chalfé, e Roger Y. Tsien il premio nobel per la chimica in relazione alla scoperta e allo sviluppo della GFP.

La luminescenza di GFP è un fenomeno intrinseco alla stessa proteina e non richiede substrati né enzimi, quindi questa è molto usata come marcatore nelle indagini di identificazione e localizzazione subcellulare delle proteine. In questi saggi la sequenza nucleotidica della proteina X da identificare viene fusa con la sequenza nucleotidica di GFP in un vettore di espressione che solitamente è un plasmide. Il plasmide ricombinante viene inserito nella cellula eucariotica, solitamente tramite elettroporazione, in modo che, una volta all'interno della cellula, il plasmide si integri nel genoma per ricombinazione sito specifica all'interno del gene per la proteina mitocondriale. Questo processo porta all'inattivazione del gene selvatico per X e all'espressione del gene ricombinante per X-GFP. L'identificazione e localizzazione subcellulare di X-GFP può allora essere visualizzata tramite la microscopia a fluorescenza che rivela i segnali prodotti da GFP e che quindi identificano anche X. Questa tecnica è molto potente in quanto la localizzazione della proteina studiata viene rivelata in vivo ed è possibile seguirla anche nel tempo.

1.2.3 Misure quantitative

L'emissione di luce da atomi o molecole può essere utilizzata per quantificare l'ammontare della molecola nel campione. La relazione tra l'intensità di fluorescenza e la concentrazione dell'analita è conseguenza diretta della legge di Lambert-Beer ed è espressa come:

$$F = k \cdot QE \cdot P_0 \cdot (1 - 10^{-\epsilon \cdot b \cdot c})$$

dove F è l'intensità di fluorescenza, k è un fattore geometrico dello strumento, QE è l'efficienza quantica (fotoni emessi/fotoni assorbiti), P_0 è la potenza radiante della sorgente d'eccitazione, ϵ è il coefficiente di estinzione molare, b è la lunghezza del percorso ottico e c è la concentrazione dell'analita.

La formula può essere approssimata a patto che le concentrazioni siano basse ($< 10^{-5} M$) e mostra una proporzionalità diretta tra l'intensità e la concentrazione dell'analita:

$$F = k \cdot QE \cdot P_0 \cdot (2.303 \cdot \epsilon \cdot b \cdot c)$$

I parametri k e QE possono essere determinati con una calibrazione dello strumento mentre ϵ e b sono noti, essendo ϵ caratteristico della specie e b definito dallo strumento. La misura quantitativa però è affetta dalle stesse limitazioni che presenta la legge di Lambert-Beer, è inoltre soggetta all'eccessivo assorbimento della radiazione di eccitazione (pre-filter effect) ed all'auto assorbimento della fluorescenza (post-filter effect).

1.3 Strumentazione e problemi annessi

Questa sezione non compie solamente una descrizione della strumentazione utilizzata ma ne presenta anche il modello di funzionamento e eventualmente le motivazioni di utilizzo.

1.3.1 Microscopio

Il microscopio a fluorescenza è un microscopio ottico utilizzato per studiare sostanze organiche e inorganiche usando il fenomeno della fluorescenza invece di (o in aggiunta a) riflessione e assorbimento. Tutti i microscopi a fluorescenza utilizzano lo stesso principio: un campione viene illuminato con una luce di una lunghezza d'onda che ne causa la fluorescenza. La luce emessa viene poi rilevata tramite l'obiettivo del microscopio. Solitamente vengono utilizzati due filtri in questa tecnica: un filtro di illuminazione (o eccitazione) che assicura che l'illuminazione sia monocromatica e ad una corretta lunghezza d'onda; un secondo filtro di emissione (o rilevamento) che assicura che nessun fascio di luce della sorgente raggiunga il rivelatore[17].

I componenti tipici di un microscopio a fluorescenza sono una fonte luminosa (e.g. lampada ad arco allo xeno, lampada ai vapori di mercurio), un filtro di eccitazione, uno specchio diroico e il filtro di emissione. I filtri e lo specchio devono essere scelti per combaciare con lo spettro di eccitazione e di emissione del fluoroforo. La maggior parte dei microscopi usati in questo campo sono detti microscopi ad epifluorescenza, questo perché sia il fascio entrante che quello uscente sono al di sopra del piano su cui è presente il fluoroforo.

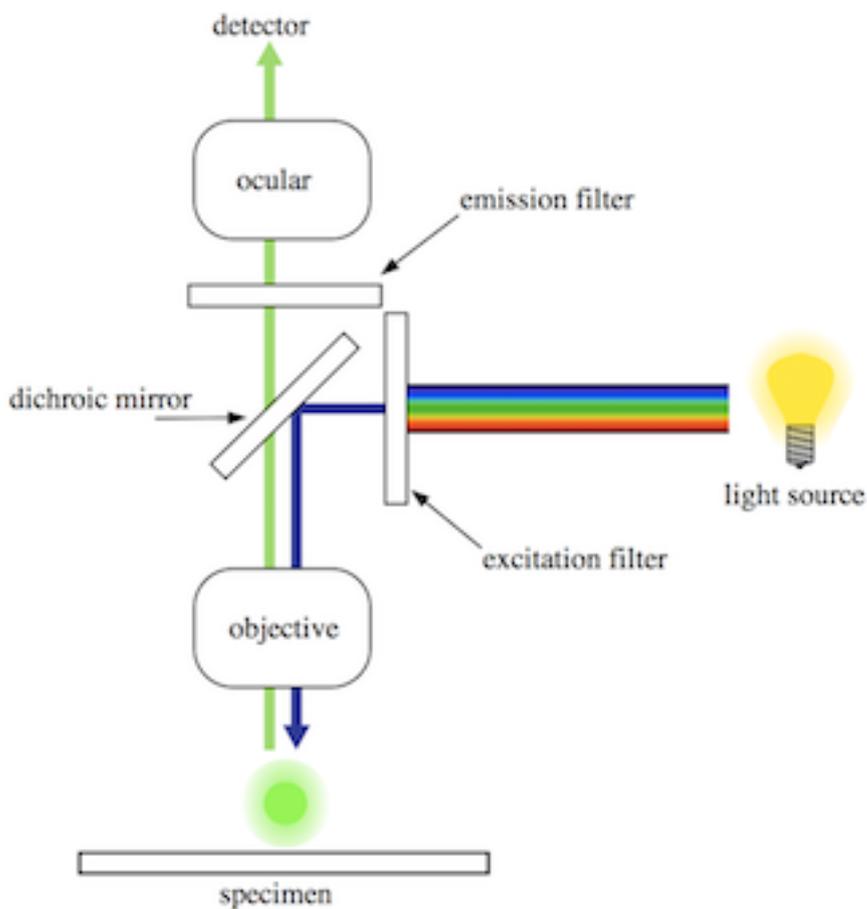


Figura 1.1: Stilizzazione di un microscopio ad epifluorescenza

Il microscopio utilizzato per le acquisizioni è il microscopio invertito Nikon TE-2000U, questo è equipaggiato con un particolare meccanismo lungo il percorso della luce che assorbe totalmente le luci di disturbo nella traiettoria ottica. Il segnale in uscita può essere deviato sugli oculari oppure su una delle altre quattro porte disponibili sul microscopio. Su una di queste porte è stata installata una telecamera con cui si farà l'acquisizione. Come sorgente luminosa viene utilizzata una lampada allo xenon da 75 Watt (lampada ad arco), questa è composta da due elettrodi di grafite che vengono inizialmente messi in contatto e poi separati per creare l'arco.

In ingresso al microscopio è necessaria una luce monocromatica, così in cascata alla sorgente luminosa viene collegato un monocromatore, ovvero un dispositivo che data in ingresso una luce policromatica ne fornisce in uscita una monocromatica. Il fascio di luce in uscita può essere controllato dall'utente e può variare da 250nm a 650nm in base alle necessità. Da quest'ultimo esce una fibra ottica che porta il fascio luminoso direttamente dentro il microscopio.

1.3.2 Fotocamera

La fotocamera utilizzata per l'acquisizione è una Nikon DS-5m dotata di un controller DS-U1, è collegata al microscopio mediante un connettore "passo C". La fotocamera viene gestita da un controller che è collegato ad essa tramite una connessione seriale e può essere collegata ad un elaboratore tramite un cavo USB. Insieme alla telecamera viene fornito anche un software (X-Pro) per l'acquisizione ma in questo caso specifico è stato progettato un software specifico che si interfaccia con i driver della fotocamera. La fotocamera ha un *CCD* da 2/3 pollici che fornisce 5.24 Mpixel dichiarati (effettivi 5.07 Mpixel), la discretizzazione massima dei valori in uscita è a 12-bit e fornisce la possibilità di acquisire immagini con un'esposizione nel range [1/1000s, 60s]. La dimensione delle immagini in uscita può essere 2560x1920, 1280x960 oppure 640x480, le prove condotte per la tesi sono state effettuate acquisendo immagini di dimensione 1280x960. Il valore di apertura nominale è $f/5.6$ o superiore. Per l'*image analysis* è molto importante conoscere gli strumenti con cui si lavora per poter predire e correggere eventuali errori dovuti ad essi, di seguito vengono analizzate due delle parti che compongono principalmente una fotocamera e i problemi che causano.

Ottica

Quando si acquisisce un'immagine il primo stage è quello che riguarda l'ottica, ovvero tutto ciò che riguarda l'apertura, lo shutter, l'esposizione, la lente e il piano focale (anche se in realtà in alcuni casi lo shutter viene simulato nello stage successivo). L'apertura di una fotocamera è definita come il rapporto tra la lunghezza focale (cioè la

distanza tra la lente e il punto di fuoco) e il diametro della pupilla di entrata. Spesso questo valore è denotato come $f/\#$, più alto è il valore è, meno luce per unità di area riceve la fotocamera, mentre raddoppiando il valore di apertura, il tempo necessario di esposizione per avere la stessa quantità di luce sul sensore quadruplica. Con l'apertura viene anche controllata la profondità di campo, riducendo l'apertura la profondità di campo aumenta.[6] L'esposizione è il tempo in cui il sensore (o la pellicola) viene irradiato dalla luminosità della scena, qualitativamente più il tempo di esposizione sarà basso e più l'immagine acquisita sarà scura poiché una quantità minore di fotoni riuscirà a colpire il sensore. Questo parametro viene gestito attraverso la velocità con cui si apre e si chiude lo *shutter*.

L'esposizione corretta per una scena è quella che riesce ad utilizzare il range dinamico più vasto, se l'esposizione è troppo elevata (*sovraesposizione*) si avranno delle zone in saturazione verso i valori maggiori, se l'esposizione è troppo bassa (*sottoesposizione*) si avranno delle zone in saturazione verso i valori minori.

Un altro parametro importante della camera è il *field of view* (o angle of view), questo determina la parte di scena che può essere catturata durante l'acquisizione. Questo parametro non è da confondere con l'angolo di copertura che descrive l'angolo di proiezione della lente sul piano focale (i.e. il sensore), per la maggior parte delle fotocamere si può supporre che il cerchio immagine prodotto dalle lenti sia largo abbastanza per coprire il sensore completamente.

Ogni ottica ha un problema intrinseco dovuto alla natura stessa della luce: il decadimento. Il decadimento della luce è proporzionale a $\cos^4 \theta$, dove θ è l'angolo con cui la luce incide sul sensore, questo porta ad un inevitabile diminuzione della luminosità della foto sui bordi del *field of view*. Con la conoscenza dei parametri sopra citati possiamo predire il risultato fisico desiderato dell'acquisizione: l'irradianza E dell'immagine è correlata alla radianza della scena L tramite la formula:

$$E = L \frac{\pi}{4} \left(\frac{d}{h}\right)^2 \cos^4 \phi$$

dove h è la lunghezza focale, d è il diametro di apertura e ϕ è l'angolo sotteso dal raggio principale sull'asse ottico. In un sistema ideale

l'intensità luminosa registrata dovrebbe essere

$$I = E \cdot t$$

dove t è il tempo di esposizione[7].

CCD

Il sensore ottico che acquisisce fisicamente l'immagine dalla scena nella fotocamera è il secondo stage durante l'acquisizione di un immagine. Nella fotocamera utilizzata questo compito è svolto da un *CCD*.

Il *CCD* (Charge-Coupled Device) è un dispositivo per il movimento di cariche elettriche, solitamente dal dispositivo stesso verso un'area in cui la carica può essere manipolata, per esempio per la conversione in valori digitali. Generalmente questo dispositivo ha la forma di una riga o una matrice. Fornendo la giusta tensione ad ogni cella della matrice o della riga è possibile far spostare le cariche in una cella adiacente.

Il *CCD* è la tecnologia maggiormente utilizzata per il *digital imaging*. Quando l'acquisizione dell'immagine inizia, questi semiconduttori vengono polarizzati sopra la soglia di inversione, permettendo la conversione dei fotoni che arrivano in cariche elettriche nell'interfaccia semiconduttore-ossido; il *CCD* viene poi utilizzato per leggere queste cariche.

Nei *CCD* utilizzati per catturare le immagini è presente una regione fotoattiva (uno strato di silicio epitassiale) e una regione trasmissiva composta da uno shift register (i.e il *CCD*). Un'immagine viene proiettata attraverso una lente nell'array di condensatori, in questo modo ogni condensatore accumula una carica proporzionale all'intensità luminosa da cui è stato investito. A questo punto l'array di due dimensioni che forma il sensore ottico è carico in modo proporzionale all'immagine proiettata sul piano focale. Una volta che il *CCD* è stato esposto all'immagine un circuito di controllo fa traslare le cariche presenti in ogni linea verso la linea adiacente, sull'ultima linea è presente un amplificatore che converte i valori di carica in tensione. Ripetendo il processo per la dimensione dell'array tutte le cariche vengono trasferite, quindi l'immagine è ora rappresentata come livelli di tensione. In un dispositivo digitale questi livelli di tensione vengono campionati e digitalizzati. I *CCD* possono essere implementati

in diverse architetture, le più comuni sono *full-frame*, *frame-transfer*, *interline*, la differenza sta nel modo in cui risolvono il problema dello *shuttering*.

Nel primo caso tutta l'area del *CCD* è attiva e non è presente alcuno shutter elettronico, quindi è necessario uno shutter meccanico.

Nel secondo caso metà dell'area è coperta con una maschera opaca (tipicamente alluminio), l'immagine può essere velocemente trasferita dalla parte attiva a quella opaca con una perdita accettabile. Questa architettura richiede un'area grande il doppio di quella precedente.

Nell'ultimo caso la parte opaca viene estesa a tutte le colonne, così viene diminuita di molto l'area fotoattiva, in questo caso la velocità di shutter è molto elevata ma è necessario porre delle microlenti sopra ogni cella del *CCD*.

Generalmente le fotocamere utilizzano una maschera Bayer sopra i *CCD*. Questa maschera consiste nel filtrare lo spettro di luce incidente nelle componenti *RGB* (Rosso, Verde e Blu). In ogni quadrato composto da 4 pixel uno è filtrato con il rosso, uno con il blu e due con il verde, questo perché l'occhio umano è molto più sensibile alle variazioni di verde che a quelle delle altre due bande. In realtà questo è un metodo "antico", ora per prestazioni migliori si preferiscono usare tre sensori (uno per ogni colore) con un prisma tricroico che segmenta lo spettro luminoso.

Vignetting

Uno dei problemi che affliggono l'acquisizione di immagini è il vignetting, con questo termine si identifica il crearsi di una zona ai bordi dell'immagine con luminosità minore (a volte completamente scura). Generalmente ciò è dovuto al primo stage della fotocamera, ovvero ciò che riguarda la parte ottica. Nonostante i produttori di lenti cerchino di minimizzare l'effetto del vignetting, è ancora presente in tutte le lenti e può essere piuttosto visibile per certe aperture e lunghezze focali.

Il vignetting può essere causato dalle seguenti 4 sorgenti:

Natural vignetting si riferisce al decadimento radiale dovuto alla geometria dell'ottica: regioni differenti del piano di immagine ricevono irradianza differente come già accennato nella sezione

1.3.2 Ottica. Non in tutte le lenti, la distanza dalla pupilla di uscita del piano immagine cambia quando la distanza di fuoco viene cambiata, quindi questa componente varia con la distanza di fuoco. La legge del \cos^4 è solo un'approssimazione e non modella bene le fotocamere reali e le lenti.

Pixel vignetting si riferisce al decadimento dovuto alla sensibilità angolare dell'ottica digitale. Questo tipo di vignetting - che affligge solo le fotocamere digitali - è dovuto alla finita profondità delle celle nel sensore digitale, ciò causa una parziale occlusione da parte dei bordi delle celle dei fotoni che incidono con un angolo molto acuto sulla superficie del sensore.

Optical vignetting si riferisce al decadimento radiale causato dal bloccaggio del percorso della luce dentro il corpo della lente da parte del diaframma. È anche conosciuto come vignetting fisico o artificiale. Questo fenomeno viene facilmente osservato cambiando l'apertura del diaframma che riduce o aumenta la quantità di luce che raggiunge il piano immagine. Il vignetting ottico è una funzione dell'apertura, può essere ridotto chiudendo l'apertura poiché una piccola apertura limita il passaggio di luce in ugual misura al centro e nei bordi del frame.

Mechanical vignetting si riferisce al decadimento radiale dovuto all'occlusione di alcuni passaggi della luce da parte di altri elementi della fotocamera, generalmente filtri o paraluce attaccati nella parte frontale della lente.

Generalmente, in tutti questi casi, il vignetting aumenta con l'apertura e diminuisce con la lunghezza focale.[5]



Figura 1.2: Sinistra: immagine di un muro a $f/1.4$. Centro: lo stesso muro a $f/5.6$, dimostra che il vignetting ottico diminuisce con la dimensione dell'apertura. Destra: la forma della pupilla di entrata varia sia con l'apertura di entrata che con l'angolo di incidenza. [5]

Funzione di risposta

Un sensore come il CCD è progettato per produrre segnali elettrici direttamente proporzionali a I (luminosità della scena registrata). Sfortunatamente ci sono altri stage nell'acquisizione di immagini che introducono non-linearità come ad esempio la conversione ADC. In questo momento le fonti particolari che generano non-linearità non sono molto rilevanti. Quello che è necessario conoscere è che la misurazione finale della luminosità M prodotta da un sistema di acquisizione è correlata alla radianza della scena I attraverso la funzione di risposta g (i.e. $M = g(I)$). Per mappare le misurazioni M sui valori scalati della radianza I , è necessario trovare la funzione inversa $f = g^{-1}$, dove $I = f(M)$. Risalire a questa funzione è il problema della *calibrazione radiometrica*. È una pratica comune stimare la funzione f mostrando al sistema una carta di calibrazione, come ad esempio la *Machbeth chart*, che include aree con riflettanza relativa conosciuta. Le radianze relative conosciute delle aree e i valori misurati vengono poi interpolati trovando un'approssimazione di f . [7]

È chiaro che questo tipo di metodo non è utilizzabile in microscopia, dato che il FOV dello strumento è troppo stretto per acquisire la carta di calibrazione.

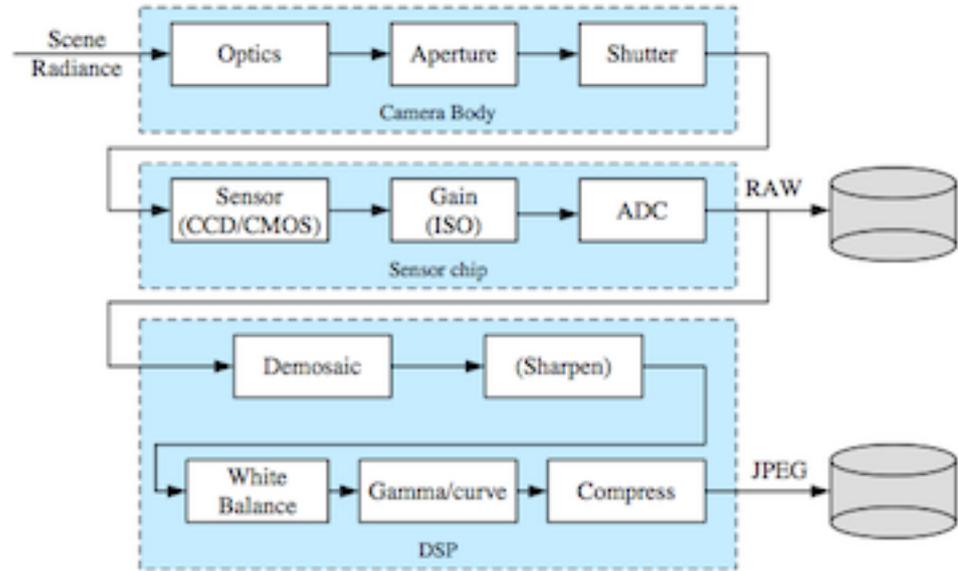


Figura 1.3: Pipeline per l'acquisizione di immagini, ognuno degli stage visualizzati introduce non-linearità o rumore [18]

1.4 Informatico

In questa sezione vengono descritte le conoscenze pregresse necessarie per ciò che riguarda il salvataggio e la lettura di immagini in un elaboratore, altre competenze nel campo dell'elaborazione delle immagini saranno coperte nel capitolo successivo.

1.4.1 Colori

Come detto nella sezione precedente, per suddividere il fascio di luce incidente nel dispositivo di acquisizione in tre colori principali - Rosso, Verde e Blu (*RGB*) - viene utilizzato un prisma e filtri tricromatici oppure una maschera di *Bayer*.

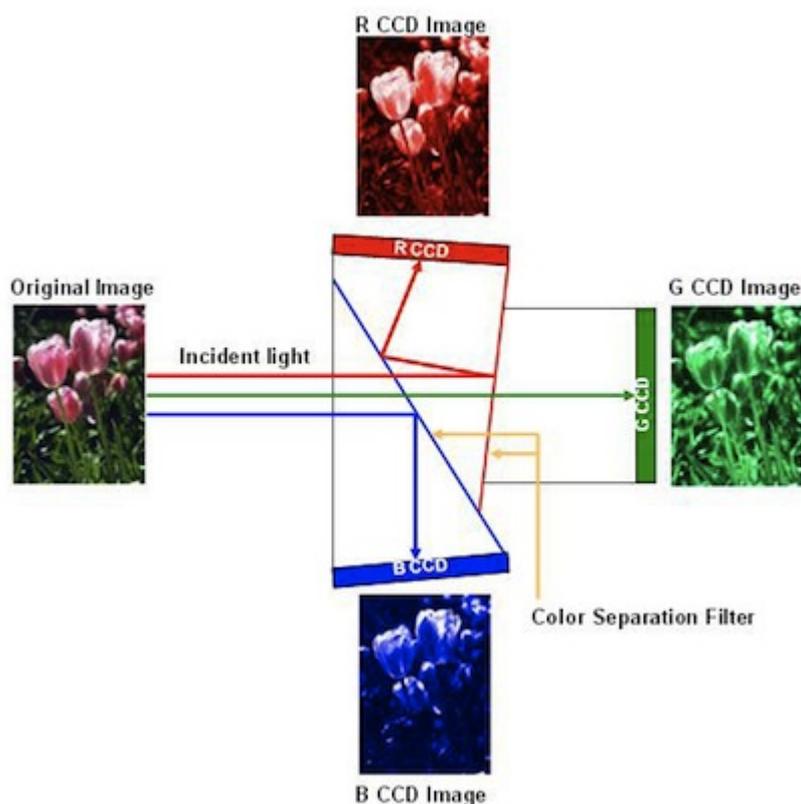


Figura 1.4: Suddivisione del fascio di luce nei tre colori fondamentali additivi per mezzo di filtri tricoici.

RGB è un modello di colori additivo, ognuno dei fasci viene chiamato componente del colore e può avere una intensità indipendente dagli altri. Si dice che il modello di colori è additivo nel senso che i tre spettri dei colori vengono sommati per creare lo spettro del colore finale.[9]

Intensità nulla per ogni colore dà il colore più scuro (nero), invece intensità massima per tutte le componenti dà il bianco. Quando l'intensità di tutti i componenti è la stessa si ha come risultato il grigio, più chiaro o più scuro in base all'intensità.

La scelta dei colori primari è correlata alla fisiologia dell'occhio umano; i colori primari buoni sono quelli che massimizzano la differenza di risposta delle cellule cono della retina umana sottoposte a luce con

lunghezza d'onda differente. Nell'occhio umano esistono tre tipi di coni e rispondono ognuno in modo migliore alle seguenti lunghezze d'onda, mediamente:

- Blu - 420nm (lunghezza d'onda corta)
- Verde - 534nm (lunghezza d'onda media)
- Rosso - 564nm (lunghezza d'onda lunga)

Il metodo con cui questi valori vengono interpolati (nel caso di un solo CCD e filtro di Bayer) dipende dall'algoritmo di *demosaiicing*.

L'*RGB* è un modello di colori device-dipendente, differenti dispositivi catturano o riproducono i valori *RGB* in modo differente. Questo avviene poiché ogni dispositivo risponde in modo differente ai singoli valori delle componenti, i livelli variano da produttore a produttore, spesso anche nello stesso dispositivo. Per avere lo stesso colore su tutti i dispositivi è necessario implementare una gestione del colore (e.g. *ICC* International Color Consortium), ovvero associare un profilo di colore ad ogni dispositivo che ne indichi le caratteristiche cromatiche.

Un colore nel modello *RGB* viene descritto indicando la quantità di ogni componente necessaria, è espresso come una tripletta (r,g,b) . Esistono principalmente quattro metodi per rappresentare un colore:

- La notazione aritmetica indica ogni componente con valori compresi in $[0,1]$, questa viene utilizzata nei sistemi che utilizzano la rappresentazione con floating-point.
- La notazione percentuale indica ogni componente con una percentuale da 0% a 100%.
- La notazione a 8-bit permette di specificare ogni componente come numero intero nel range $[0, 255]$, oppure come numero esadecimale $[00, FF]$.
- La notazione a 16-bit permette di specificare ogni componente come numero intero nel range $[0, 65535]$.

Spesso questi valori non vengono utilizzati come lineari ma sono sottoposti ad una correzione di gamma dopo l'acquisizione.

Esistono anche altre codifiche per i colori (e.g. *HSI*, *CMY*) utilizzate in altri ambiti ma per lo scopo della tesi non è necessario descriverle.

1.4.2 Formato dell'immagine

Il formato utilizzato per salvare le immagini dal microscopio è il *Tagged Image File Format* (TIFF). Questo è estremamente indicato per questo campo poiché ha la capacità di salvare sia immagini compresse che immagini non compresse. Nel nostro caso è molto importante avere immagini non compresse poiché causerebbero una perdita di dati. Le immagini utilizzate, in particolare, sono non compresse con una codifica dei colori a 24-bit, ovvero 8-bit per ogni canale.

Capitolo 2

Tecniche di elaborazione delle immagini

In questo capitolo verranno descritte tutte le strategie, le metodologie e gli algoritmi necessari per lo sviluppo del software.

La maggior parte di questi algoritmi fanno parte dell'ambito di *image processing*, ovvero una branca dell'elaborazione delle immagini che si occupa di processare immagini per risaltarne particolari, togliere il rumore, migliorarne le caratteristiche, L'*image processing* prende come input un'immagine e la restituisce in output modificata, questo si discosta dall'ambito di *image analysis* in cui si prende in ingresso un'immagine e si forniscono in uscita delle misure riguardanti l'immagine stessa. Generalmente per avere una buona analisi dell'immagine è necessario prima sottoporla ad un processo di elaborazione. Nelle descrizioni seguenti - ove non specificato - verranno usati algoritmi studiati per lavorare su immagini *greyscale*, ciò è dovuto al fatto che la fluorescenza dei batteri si manifesta principalmente nelle medie lunghezze d'onda, quindi dell'immagine acquisita viene utilizzato solo il canale verde.

2.1 Tipologie di immagini

Le immagini trattate nell'applicazione sono principalmente di tre tipi:

A Colori sono immagini a 24-bit o a 32-bit e sono rappresentate con una codifica *RGB* come spiegato nella sezione 1.4.1. Nelle immagini a 32-bit gli 8-bit più significativi sono destinati all'alpha, ovvero alla trasparenza di un determinato pixel.

Greyscale sono immagini con un solo canale, quando un'immagine a colori viene suddivisa in canali singoli, ognuno di questi può essere considerato come un'immagine greyscale. La profondità del canale dipende dalla precisione di cui si necessita, scindendo un'immagine a colori a 24-bit si avranno tre immagini greyscale da 8-bit ognuna.

Binarie sono immagini logiche, in cui ogni pixel può avere solo i valori $\{0,1\}$, generalmente vengono usate durante la segmentazione.

2.2 Convoluzione

La convoluzione è un'operazione che viene spesso utilizzata nell'*image processing*, è quindi necessario introdurlo prima di affrontare l'argomento.

Questa operazione permette di “moltiplicare insieme” due array di numeri, generalmente di dimensioni differenti ma della stessa dimensionalità (cioè matrici con matrici, ...) per produrre un terzo array di numeri della stessa dimensionalità. La convoluzione può essere usata per implementare operatori il quale output è un pixel combinazione lineare di certi valori di pixel di input.

Nel contesto dell'*image processing*, uno degli array in input generalmente è una immagine greyscale. Il secondo array generalmente è molto più piccolo, ed è anche questo di due dimensioni, è conosciuto con il nome di *kernel*. Date le due matrici seguenti, in cui quella a sinistra è l'immagine di input e quella a destra il kernel

$$\begin{array}{c}
 \left[\begin{array}{ccccc}
 I_{11} & I_{12} & I_{13} & I_{14} & I_{15} \\
 I_{21} & I_{22} & I_{23} & I_{24} & I_{25} \\
 I_{31} & I_{32} & I_{33} & I_{34} & I_{35} \\
 I_{41} & I_{42} & I_{43} & I_{44} & I_{45} \\
 I_{51} & I_{52} & I_{53} & I_{54} & I_{55}
 \end{array} \right] \\
 \text{Immagine}
 \end{array}
 \quad
 \begin{array}{c}
 \left[\begin{array}{ccc}
 K_{11} & K_{12} & K_{13} \\
 K_{21} & K_{22} & K_{23}
 \end{array} \right] \\
 \text{Kernel}
 \end{array}$$

la convoluzione viene eseguita “sovrapponendo” il kernel all’immagine, generalmente partendo dall’angolo in alto a sinistra, così da muovere il kernel attraverso tutte le posizioni in cui è completamente contenuto nei limiti dell’immagine (Implementazioni diverse si comportano diversamente sui bordi dell’immagine). Ogni posizione del kernel corrisponde ad un solo pixel di output, il valore del quale è calcolato moltiplicando il valore del kernel con il corrispondente valore dell’immagine (quello che gli sta sotto) per ogni cella del kernel. I valori ottenuti dalla moltiplicazione di ogni cella del kernel vengono sommati[4]. Di seguito l’esempio del calcolo della posizione in alto a sinistra:

$$O_{11} = I_{11}K_{11} + I_{12}K_{12} + I_{13}K_{13} + I_{21}K_{21} + I_{22}K_{22} + I_{23}K_{23}$$

2.3 Filtraggio

Spesso le immagini provenienti da un’acquisizione portano con sè del rumore, dovuto in parte alla scena e in parte al sistema di acquisizione, ciò disturba e può indurre gli algoritmi a dare risultati sbagliati, per questo è sempre opportuno fare un filtraggio prima di sottoporre l’immagine ad un *image processor*. Il metodo di filtraggio più semplice è quello di compiere una convoluzione sull’immagine con un kernel media, ovvero assegnare ad ogni pixel il valore della media dei propri vicini. Si è notato che questo metodo di filtraggio è piuttosto efficace anche se tende ad “appiattire” l’immagine. Una tecnica molto più efficace consiste nel calcolare una media pesata dei vicini, dando più peso ai vicini adiacenti e meno peso ai vicini più lontani, in questo modo si forma un *kernel gaussiano*.

Il *kernel gaussiano* (usato appunto nel filtro omonimo) viene costruito

con la seguente formula:

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

dove x è la distanza dall'origine nell'asse orizzontale, y la distanza dell'origine nell'asse verticale e σ la deviazione standard della distribuzione gaussiana (Per la costruzione di un kernel si considera come origine la cella posta al centro) [4]. Generalmente il valore di σ si sceglie come

$$\sigma = \frac{K + 1}{6}$$

dove K è il lato del kernel che si vuole utilizzare.

Nello svolgimento della tesi questo filtro verrà utilizzato prima di ogni elaborazione applicata sull'immagine sorgente.

2.4 Segmentazione

La segmentazione è quel processo con cui l'immagine viene partizionata in vari segmenti (insiemi di pixel). L'obiettivo della segmentazione è semplificare e/o cambiare la rappresentazione di un'immagine in qualcosa che sia più significativo e facile da analizzare. Questo procedimento viene spesso utilizzato per localizzare oggetti e bordi nelle immagini. Più precisamente, la segmentazione è quel processo in cui si assegnano delle etichette ad ogni pixel di un'immagine in modo che i pixel con la stessa etichetta condividano certe caratteristiche visuali. Il risultato della segmentazione è un insieme di segmenti che collettivamente coprono tutta l'immagine (o un insieme di contorni nel caso di *edge detection*), i pixel di ogni segmento sono simili ai vicini rispetto ad alcune caratteristiche mentre quelli delle regioni adiacenti sono significativamente differenti rispetto le stesse caratteristiche.

Nell'elaborazione delle immagini, la segmentazione di immagini è uno dei problemi più ampio e più studiato. Tecniche più antiche tendono ad usare l'unione o la divisione di regioni, questi - nella letteratura del clustering - corrispondono ad algoritmi *divisivi* o *agglomerativi*. Algoritmi più recenti spesso ottimizzano qualche criterio globale come la consistenza intra-regione, la lunghezza di confine inter-regione o dissimilarità.[18] Esistono molti altri tipi di algoritmi come quelli

basati sugli istogrammi, gli edge detector, algoritmi basati sui contorni attivi, algoritmi basati sui grafi, region-growing, Di seguito è proposta solo la trattazione degli algoritmi che verranno poi utilizzati: di *sogliatura* e *edge detection*.

2.4.1 Sogliatura

La sogliatura è un metodo di binarizzazione dell'immagine (se effettuato con una sola soglia) che consiste nel definire una soglia per dividere l'immagine in due regioni principali, gli oggetti con valore al di sopra della soglia (*foreground*) e quelli con valore al di sotto (*background*). Questo metodo può essere suddiviso in due grandi categorie: la *sogliatura globale* e la *sogliatura locale*. Il primo consiste nel definire una sola soglia per tutta l'immagine, questo però può portare dei problemi nel caso l'immagine sia affetta da una illuminazione non uniforme oppure nel caso sia presente del rumore sull'immagine. Uno degli algoritmi più conosciuti ed efficaci di questo genere è *Otsu* che posiziona la soglia nella zona in cui il taglio è più efficace se si ha una distribuzione con due creste. La seconda categoria definisce dei metodi in cui l'elaborazione avviene *per-pixel*, ovvero viene processata l'immagine e si definisce una soglia per ogni pixel, funzione delle caratteristiche dei vicini del pixel stesso.

La *sogliatura locale* (detta anche sogliatura adattiva) permette di risolvere (dipendendo dall'entità) il problema dell'illuminazione non uniforme e del *vignetting* nella fase di segmentazione, per questo motivo è stato scelto come metodo di sogliatura iniziale. Di questi metodi ne sono stati presi in analisi principalmente tre che verranno esposti di seguito.

Sogliatura adattiva generica

Questo approccio consiste nel trovare una soglia locale per ogni pixel esaminando statisticamente i valori di intensità dei vicini. La statistica più appropriata dipende dall'immagine sottoposta. Statistiche che si possono usare sono la media, la mediana o la media tra il valore massimo e il valore minimo. La grandezza del vicinato deve essere larga in modo tale da coprire abbastanza pixel del *foreground* e del

background, altrimenti verrà scelta una soglia non ottimale. In modo opposto se viene scelto un vicinato troppo largo si viola l'assunzione che l'illuminazione sia approssimativamente uniforme.

Questo metodo può essere migliorato utilizzando una costante sottrattiva rispetto alla soglia calcolata con il procedimento sopra descritto.

Tecnica di Sauvola

Data un'immagine greyscale in cui $g(x, y) \in [0, 255]$ è l'intensità di un pixel posizionato in (x, y) , l'obiettivo è trovare una soglia tale che il pixel corrispondente della binarizzazione o sia:

$$o(x, y) = \begin{cases} 0 & g(x, y) \leq t(x, y) \\ 255 & otherwise \end{cases}$$

Con il metodo di binarizzazione di Sauvola la soglia $t(x, y)$ viene calcolata usando la media $m(x, y)$ e la deviazione standard $s(x, y)$ delle intensità dei pixel in una finestra $w \times w$ centrata nel pixel (x, y) :

$$t(x, y) = m(x, y) \left[1 + k \left(\frac{s(x, y)}{R} - 1 \right) \right]$$

dove R è il valore massimo per la deviazione standard ($R = 128$ per un'immagine greyscale), e k è un parametro che prende valori positivi nel range $[0.2, 0.5]$. La media locale $m(x, y)$ e la deviazione standard $s(x, y)$ adattano il valore della soglia con il contrasto nel vicinato del pixel. Quando c'è un contrasto alto in alcune regioni dell'immagine, $s(x, y) \approx R$ che risulta in $t(x, y) \approx m(x, y)$. Questo è lo stesso risultato che si ha con il metodo di *Niblack's*. La differenza sorge quando il contrasto nel vicinato è piuttosto basso. In quel caso la soglia va al di sotto del valore della media quindi rimuovendo con successo le parti scure del *background*. Il parametro k controlla il valore della soglia nella finestra locale in modo che più alto è il valore di k e più bassa è la soglia dalla media.[14]

Algoritmo di risonanza

La teoria della risonanza assume che in una scena ogni punto abbia massa m . Questi punti non sono isolati dagli altri ma connessi da

inter-forze. Siccome i punti non sono isolati, il movimento di un punto risulta nella diffusione del proprio effetto agli altri punti che sono attorno ad esso.

Se si assume che un punto è la sorgente della risonanza e un altro punto ne può essere largamente affetto allora la loro distanza deve necessariamente essere piuttosto piccola e anche la differenza tra la frequenza esterna e quella naturale deve esserlo. Queste due condizioni possono essere soddisfatte se una soglia è posta in modo che la differenza tra la soglia esterna e quella naturale sia abbastanza piccola, e che l'algoritmo di risonanza venga utilizzato tra punti adiacenti per assicurare che la distanza tra essi sia abbastanza piccola.

Diffondendo la risonanza i punti adiacenti che hanno le stesse caratteristiche sono clusterizzati in una sola regione. Questo processo sembra simile all'algoritmo generale di *region growing* ma sono essenzialmente diversi. Questo algoritmo enfatizza la similarità tra i punti adiacenti, la risonanza può essere diffusa da punto a punto, in questo modo può essere risolto il problema causato dalle gradazioni di intensità. Solo cambi repentini di caratteristiche tra punti adiacenti possono creare un confine di regioni diverse.

Sia definita la relazione $P_\delta(a, b)$ come il percorso tra il punto a e il punto b (non è necessario che i punti siano adiacenti) tramite la soglia δ (valore per stimare la differenza di caratteristiche tra due punti adiacenti). Se c'è una sequenza di punti adiacenti che connettono a con b , tali che abbiano le stesse caratteristiche (i.e. valore di intensità) o caratteristiche lievemente differenti (comunque sotto la soglia δ), allora a e b fanno parte della stessa regione. Se s è un punto sull'immagine, tutti i punti x_i che soddisfano la relazione $P_\delta(s, x_i)$ formeranno la regione $R_\delta(x, y)$. Il punto s viene chiamato *seed*.

Ora il principio dell'algoritmo di risonanza per la segmentazione è chiaro: partendo da uno o più pixel *seed*, i pixel adiacenti che appartengono alla stessa regione sotto δ vengono clusterizzati, ciò avviene fino quando tutti i pixel sono stati processati. La scelta iniziale dei *seed* non influenza la segmentazione risultante. [3] Un vantaggio di questo algoritmo è che si risparmia la fase di *labeling* invece necessaria nei due metodi precedenti.

2.5 Edge detection

L'*edge detection* è uno strumento essenziale nell'elaborazione delle immagini, punta a identificare i punti dell'immagine nei quali si ha una brusca variazione di intensità. Ci sono vari metodi per completare questo compito, ma la maggior parte di questi sono raggruppabili in due categorie: *search-based* e *zero-crossing based*. I metodi *search-based* scovano i bordi prima calcolando una misura della "durezza" del bordo - di solito con una derivata di prim'ordine come la magnitudo del gradiente - e poi cercando la direzione del bordo, utilizzando per esempio la direzione del gradiente. I metodi basati su *zero-crossing* ricercano l'attraversamento dello zero in una derivata di secondo ordine come per esempio il *Laplaciano*.

Per l'applicazione in esame è stato considerato come edge detector il *Canny Operator* poichè ha dato buoni risultati.

2.5.1 Sobel operator

Questo operatore è uno dei possibili che vengono utilizzati per calcolare le derivate direzionali del gradiente di un immagine partendo dal calcolo della derivata in direzione x e in direzione y . Per fare ciò si utilizzano due *kernel* di convoluzione, l'uno ruotato di 90° rispetto l'altro. Di seguito a sinistra il kernel utilizzato per trovare le derivate rispetto x e a destra quello utilizzato per calcolare le derivate rispetto ad y .

$$\begin{bmatrix} -1 & 0 & +1 \\ -2 & 0 & +2 \\ -1 & 0 & +1 \end{bmatrix} \quad \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Ottenute le due matrici con le derivate direzionali (G_x e G_y) si possono ottenere la matrice della magnitudo del gradiente in ogni punto e la matrice della direzione del gradiente in ogni punto. La prima viene calcolata eseguendo pixel per pixel l'operazione

$$|G| = \sqrt{G_x^2 + G_y^2}$$

la seconda matrice invece viene calcolata eseguendo pixel per pixel l'operazione

$$\theta = \arctan\left(\frac{G_y}{G_x}\right)$$

ma è necessario stare attenti poiché la funzione \arctan restituisce solo valori nel range $[-\frac{\pi}{2}, \frac{\pi}{2}]$, se fosse necessario un angolo compreso nel range $[-\pi, \pi]$ allora bisognerebbe valutare anche il segno delle derivate. In generale questo operatore viene preferito rispetto al *Robert Cross Operator* poiché utilizza un kernel più ampio e quindi è meno sensibile ai rumori.

2.5.2 Canny Operator

Questo operatore è composto da tre fasi principali: la prima in cui viene calcolato il gradiente nei vari punti dell'immagine, la seconda in cui si opera una soppressione dei risultati non massimi e la terza in cui si applica una soglia con isteresi.

Per trovare il gradiente di intensità dell'immagine viene utilizzato l'operatore di Sobel descritto nella sezione 2.5.1, la direzione va però arrotondata ad uno dei quattro angoli rappresentanti l'asse verticale, orizzontale o una delle due diagonali (i.e. 0° , 45° , 90° , 135°).

Una volta ottenuto sia la magnitudo che la direzione del gradiente è necessario eseguire una soppressione dei non-massimi, ovvero si effettua una ricerca per determinare se la magnitudo assume dei massimi locali nella direzione del gradiente, quindi si analizza:

- se l'angolo del gradiente dopo esser stato arrotondato è 0° (i.e. il bordo è nella direzione nord-sud), il punto viene considerato sul bordo se la magnitudo del gradiente è maggiore della magnitudo dei gradienti nella direzione ovest e est,
- se l'angolo del gradiente dopo esser stato arrotondato è 90° (i.e. il bordo è nella direzione ovest-est), il punto sarà considerato sul bordo se la magnitudo del gradiente è maggiore della magnitudo dei gradienti nella direzione nord e sud,
- se l'angolo del gradiente dopo esser stato arrotondato è 135° (i.e. il bordo è nella direzione nord est-sud ovest), il punto sarà

considerato sul bordo se la magnitudo del gradiente è maggiore della magnitudo dei gradienti nella direzione nord-ovest e sud-est,

- se l'angolo del gradiente dopo esser stato arrotondato è 45° (i.e. il bordo è nella direzione nord ovest-sud est), il punto sarà considerato sul bordo se la magnitudo del gradiente è maggiore della magnitudo dei gradienti nella direzione nord-est e sud-ovest,

Da questo stage si ottiene un'immagine binaria che spesso è chiamata "bordi sottili".

Nella terza fase viene utilizzata una sogliatura con isteresi composta da due soglie, una alta e una bassa. L'assunzione che i bordi importanti dovrebbero essere curve continue nell'immagine permette di seguire una debole linea e scartare i pixel di rumore che non costituiscono la linea ma hanno prodotto un grande gradiente. Da qui si inizia ad applicare la soglia alta, marcando tutti i bordi che possiamo esser sicuri essere giusti. Con questa informazione si possono iniziare a tracciare i bordi seguendo le informazioni direzionali ottenute prima, procedendo si applica la soglia bassa permettendo di continuare a disegnare bordi fin quando si trova un punto adiacente al di sopra di quest'ultima.

2.6 Operatori morfologici

Gli operatori morfologici vengono spesso applicati dopo uno stage di sogliatura o edge detection poiché sono molto adatti a lavorare con immagini binarie. Generalmente l'immagine binaria viene utilizzata come input per questi operatori insieme ad uno *structuring element*. Un elemento strutturale è una griglia simile ad un *kernel* (visto nella sezione 2.2) che viene sovrapposto all'immagine di input, il metodo di applicazione è identico a quello della convoluzione ma l'operazione che ne viene fatta dipende dall'operatore morfologico[4]. Un operatore morfologico è definito da uno *structuring element* e dall'operazione utilizzata.

2.6.1 Hit and Miss

La trasformazione *hit-and-miss* è una operazione generale morfologica binaria che può essere usata per cercare particolari pattern di background e foreground. È di fatto l'operazione binaria morfologica di base poiché la maggior parte delle altre possono essere derivate da essa, come le altre operazioni morfologiche prende in ingresso un'immagine binaria e un elemento strutturale, produce un'altra immagine binaria in uscita.

Si può quindi dire che *hit-and-miss* è una generalizzazione degli operatori morfologici e può essere "programmato" con un qualsiasi *structuring element* e operazione.

2.6.2 Opening

L'*opening* è un'operazione morfologica che come effetto ha quello di preservare tutte le regioni di foreground che riescono a contenere l'elemento strutturale utilizzato. Viene impiegato durante la segmentazione per separare gli oggetti che sono collegati da parti molto sottili (più piccole dello *structuring element*). L'implementazione dell'*opening* consiste semplicemente nell'applicazione di un'erosione seguita da una dilatazione utilizzando lo stesso elemento strutturale.

2.6.3 Erosione

L'effetto base di questo operatore morfologico è quello di erodere pixel dei bordi delle regioni di foreground, dopo l'applicazione le aree si restringono e i buchi si allargano. Un esempio di *structuring element* utilizzato è quello quadrato di dimensioni 3×3 , questo è composto da una griglia bidimensionale di lato 3 con tutte le celle con valore 1.

Il modello di applicazione dell'operatore è lo stesso spiegato nella sezione 2.6.1, l'elemento strutturale viene posto in sovrapposizione a tutta l'immagine, in ogni iterazione se sotto ad ogni sua cella è presente un pixel di foreground allora il pixel dell'immagine binaria risultante viene lasciato al valore 1, se uno solo dei vicini (considerando una connettività 8) è background allora viene posto al valore 0.

2.6.4 Dilatazione

Questa operazione è la duale di quella precedente, quindi consiste nell'applicare lo stesso operatore al background invece che al foreground: mentre nel caso dell'erosione i pixel dello structuring element venivano confrontati con il foreground, ora vengono confrontati con il background. Per l'applicazione viene utilizzato lo stesso structuring element dell'erosione.

2.7 Labeling

Questa operazione permette di "etichettare" le regioni trovate con gli algoritmi precedentemente descritti (quindi su immagini binarie) in modo da differenziare le une dalle altre. In particolare viene analizzato l'algoritmo *Connected Component Labeling* che riesce ad "etichettare" tutte le regioni scorrendo l'immagine solo due volte.

Questa procedura consiste nello scorrere l'immagine pixel per pixel - dall'angolo in alto a sinistra - in modo da identificare tutte le regioni connesse. Si può scegliere se avere una connettività 8 (quindi con i vicini NW, N, NE, E, SE, S, SW, W) oppure solo una connettività 4 (quindi con i vicini N, S, E, W). Ad ogni passo di scansione l'algoritmo si muove lungo le righe dell'immagine e quando trova un punto p (diverso dal background) ne esamina i vicini che ha già incontrato nella scansione (con connettività 8 i vicini W, NW, N, NE - con connettività 4 i vicini W, N), basandosi sulle loro informazioni l'etichettatura del pixel corrente dipende dai seguenti casi:

- se tutti i vicini hanno valore 0 (i.e. background) si assegna una nuova etichetta a p ;
- se solo un vicino fa parte del foreground, si assegna al pixel corrente l'etichetta del vicino;
- se più di un vicino fa parte del foreground allora si assegna una delle loro etichette al pixel corrente p e si fa una nota di equivalenza;

Al termine del primo scorrimento dell'immagine alle coppie di etichette equivalenti viene assegnata una singola classe, come passo finale un

secondo scorrimento sostituisce tutte le etichette equivalenti con la nuova singola classe[4].

2.8 Minimum Enclosing Rectangle

L'algoritmo che verrà descritto non è esattamente un algoritmo di elaborazione delle immagini, deriva sicuramente dalla geometria piana, ma capita spesso di utilizzarlo in questo ambito.

La ricerca del *MER* è un'attività che viene fatta principalmente per definire la lunghezza assiale e l'area occupata da un oggetto in un'immagine. Con *MER* si intende il rettangolo che circoscrive l'oggetto in cui il lato minore è il minimo possibile. Per raggiungere questo scopo è necessario trovare il *convex-hull* dei punti facenti parte dell'oggetto di cui si vuole cercare il *MER*, questo perchè l'algoritmo di ricerca del *MER* deriva dall'osservazione che quest'ultimo deve necessariamente avere un lato collineare al *convex-hull* del poligono. Ottenuto il guscio convesso attraverso algoritmi come il *Graham scan* si possono cercare i rettangoli circoscritti con l'algoritmo *Rotating calipers* per poi identificare il minore.

Siccome la complessità degli algoritmi sopra menzionati dipende fortemente dal numero di punti coinvolti nell'oggetto, è buona norma utilizzare solo i punti (i.e. pixel) di confine dell'oggetto stesso per non appesantire la computazione.

2.9 Funzione di risposta

Come accennato nella sezione 1.3.2, le possibili non-linearità introdotte durante le fasi di formazione dell'immagine nei dispositivi di acquisizione possono indurre ad una relazione non lineare tra radianza della scena ed i corrispondenti valori di intensità dei pixel nell'immagine. Questo aspetto diviene fondamentale quando dalle immagini si vogliono fornire misure legate all'intensità di ciò che si sta acquisendo, come nel caso in esame. Per fornire una misura il più possibile legata alla scena, ed ottenere quindi valori dipendenti solo dalla risposta in fluorescenza dei batteri, occorre innanzitutto valutare e correggere la distorsione radiometrica del dispositivo. A tal fine, è necessario ricava-

re la funzione di risposta del dispositivo utilizzato per poi compensarne le eventuali non linearità in modo da riportare la misurazione in un dominio radiometrico.

Il modo più immediato per ricavare la funzione di risposta avviene effettuando una calibrazione del dispositivo di acquisizione misurando aree a riflettanza nota utilizzando apposite carte di calibrazione. Questo metodo diviene impraticabile quando, come nel caso in esame, l'acquisizione avviene mediante un microscopio, dove l'area ristretta del campo di vista non consente una rappresentazione globale della carta di calibrazione. Perciò è stato utilizzato un metodo che sfrutta alcune proprietà legate al tempo di esposizione per ricavare la funzione di risposta analizzando una sequenza di immagini della stessa scena ripresa ad esposizioni diverse. Questo metodo si basa su un lavoro di T. Mitsunaga e S. K. Nayar [7] poi esteso da Alessandro Bevilacqua, Alessandro Gherardi e Ludovico Carozza in [2].

Quando vengono acquisite delle immagini a differenti esposizioni, la luminosità misurata cambia con l'esposizione mentre la radianza della scena rimane chiaramente costante. Questa osservazione permette la stima della funzione inversa f senza avere conoscenza a priori della radianza della scena. Per fare in modo che l'algoritmo descritto in seguito sia efficace è necessario pre-processare le immagini utilizzate come input per usufruire solo delle zone in cui è assente il vignetting, il quale potrebbe portare a dei risultati sbagliati.

Il modello studiato suggerisce che ogni funzione di risposta può essere modellata come una polinomiale di ordine elevato:

$$I = f(M) = \sum_{n=0}^N c_n M^n$$

in cui l'ordine dipende chiaramente dalla funzione da stimare. Quindi, la calibrazione può essere vista come la determinazione dell'ordine N e dei coefficienti c_n .

Si considerino due immagini di una scena prese utilizzando due esposizioni differenti, e_q e e_{q+1} , dove $R_{q,q+1} = e_q/e_{q+1}$. La funzione di risposta può essere trovata formulando la funzione di errore:

$$\varepsilon = \sum_{q=1}^{Q-1} \sum_{p=1}^P \left[\sum_{n=0}^N c_n M_{p,q}^n - R_{q,q+1} \sum_{n=0}^N c_n M_{p,q+1}^n \right]^2$$

CAPITOLO 2. TECNICHE DI ELABORAZIONE DELLE IMMAGINI

dove Q è il numero di immagini utilizzate, P è il numero di pixel di un'immagine e $M_{p,q}$ è il valore del pixel p nell'immagine q . Se si normalizzano tutte le misurazioni in modo che per ogni pixel $0 \leq M \leq 1$ e si fissa $f(1) = I_{max}$ allora si ha il vincolo ulteriore:

$$c_N = I_{max} - \sum_{n=0}^{N-1} c_n$$

I coefficienti della funzione di risposta vengono determinati risolvendo il sistema di equazioni lineari che risulta imponendo

$$\frac{\partial \varepsilon}{\partial c_n} = 0$$

Il miglioramento proposto è dovuto al fatto che nella soluzione precedentemente analizzata non viene inserito alcun vincolo riguardante l'esposizione, questo può portare a due gravi conseguenze e a una funzione di risposta sbagliata. Il primo è il caso in cui si utilizzino un numero troppo basso di esposizioni, quindi si può incorrere in una funzione di risposta non monotonica, il secondo è il caso in cui si abbiano vari campioni vicini alla saturazione che quindi non portano informazioni affidabili sulla funzione di risposta.

Per risolvere questi problemi vengono aggiunti due vincoli:

- Il vincolo di monotonicità consiste nello scegliere solo le coppie che soddisfano

$$M_{p,q} < M_{p,q+1}$$

Per fare in modo che eventuale rumore non disturbi la valutazione le immagini vengono pre-filtrate con un filtro gaussiano. Di per se questo metodo non garantisce la monotonicità della funzione di risposta ma ne migliora comunque l'affidabilità.

- Vengono scartati tutti i campioni con un valore vicino alla saturazione poiché si ritiene che non portino un'informazione utile - riguardante la scena - per il calcolo della funzione di risposta.

Lo sviluppo del sistema di risoluzione e il codice *MATLAB* per la valutazione possono essere trovati nell'appendice A.

*CAPITOLO 2. TECNICHE DI ELABORAZIONE DELLE
IMMAGINI*

Capitolo 3

Progetto

In questo capitolo vengono analizzate tutte le scelte progettuali effettuate per giungere alla soluzione del problema. Nella prima parte viene analizzato il problema dal punto di vista dell'elaborazione delle immagini, nella seconda parte viene analizzato il problema dal punto di vista dell'interfaccia grafica che farà da “ponte” tra l'utilizzatore e il software di analisi. Questo capitolo è completamente “*platform-independent*”, quindi fornirà un'analisi oggettiva del problema e prescinde dalla parte implementativa.

3.1 Motore di elaborazione

In un buon processo di sviluppo del software è bene creare inizialmente dei test set con cui poter validare il codice una volta scritto. Nel campo dell'*image processing* questo risulta più complesso poiché è difficile creare delle asserzioni sulle immagini con cui fare dei test veri e propri. La strada che ho percorso è un'astrazione di questo metodo, inizialmente vengono creati dei prototipi utilizzando il software *MATLAB*, il risultato di questi verrà poi confrontato con l'implementazione vera e propria. La procedura dal punto di vista teorico è molto simile ma si differenzia nel fatto che il confronto tra i risultati delle immagini viene eseguito “manualmente”. Questo approccio può essere giustificato dal fatto che in *MATLAB* spesso sono presenti funzioni *built-in* che non possono essere riprodotte fedelmente e quindi non si può fare un confronto diretto con i risultati.

La costruzione di prototipi in tempi brevi attraverso *MATLAB* è un'ottima pratica nel *software engineering* poiché dà la possibilità di avere subito dei risultati presentabili al committente (i.e. il biologo) per capire se si sta lavorando su ciò che davvero è stato richiesto. Inoltre, utilizzando questo strumento si possono anche testare vari algoritmi scegliendo quello più adatto al caso di studio.

A fronte di queste considerazioni il lavoro di analisi del problema presentatomi è iniziato dallo studio del set di immagini acquisite (specifiche sul set si possono trovare nella sezione 3.1.4).

3.1.1 Analisi dei dati di input

Inizialmente, quando mi sono stati forniti i dati di input, mi è stato comunicato che durante l'acquisizione erano sorti dei problemi sui driver della fotocamera, alcune immagini potevano aver delle esposizioni sbagliate. Per scovare gli errori è stato necessario calcolare una media dei valori delle intensità di ogni foto per vedere quali fossero degli *outliers*. Prese due acquisizioni della stessa scena con esposizione crescente, se la seconda presenta un valor medio più basso o uguale alla prima, sicuramente va classificata come errore.

Ad una prima analisi grossolana del set di immagini che si dovranno poi elaborare si può notare come il background vari all'aumentare dell'esposizione, e quindi come sia anch'esso in minima parte fluorescente. Da questa considerazione si può escludere a priori una segmentazione con soglia fissata. Con un'analisi più attenta si può notare anche che il background non è la parte più scura dell'immagine, ma sono presenti dei batteri su cui non ha agito il *GFP* che hanno un valore di intensità minore del background stesso. Questo problema si può vedere nelle immagini 3.1. Da queste si può notare anche il modo in cui un'esposizione troppo alta porti i batteri in una zona di saturazione, rendendo le misure non affidabili.

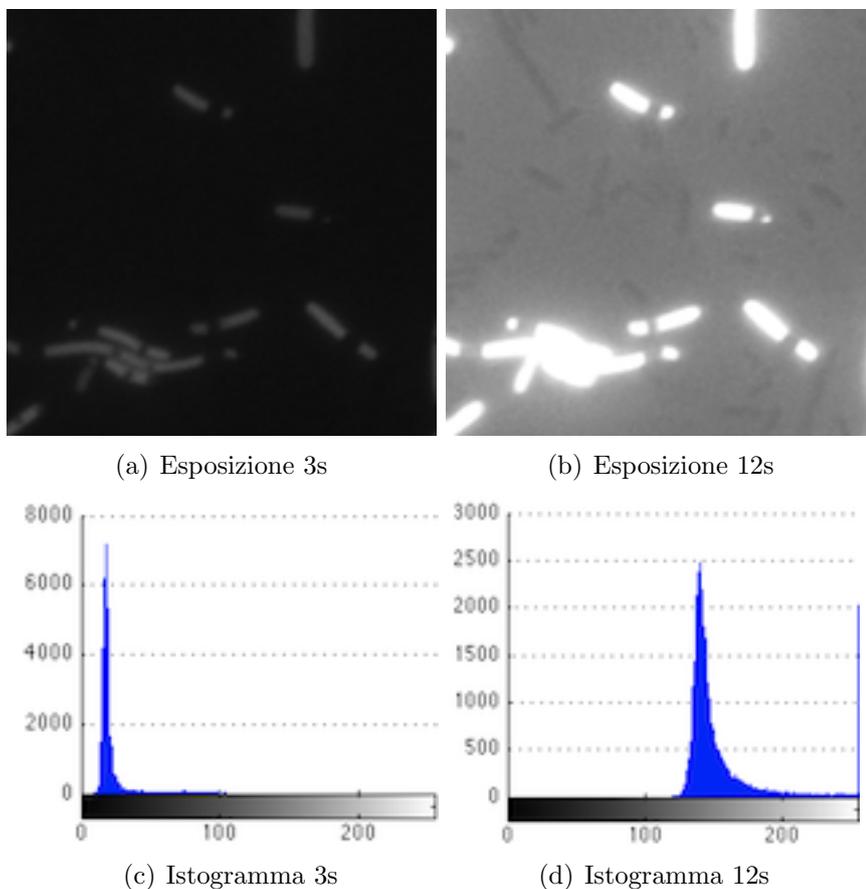


Figura 3.1: Dettaglio di microscopia in fluorescenza di batteri geneticamente modificati sottoposti al GFP, intensità del canale verde alle esposizioni specificate con rispettivo istogramma delle intensità.

Un ulteriore problema che si può notare è che certi batteri appaiono spezzati. Dopo una consulenza con l'ingegnere biomedico Filippo Piccinini è emerso che il problema può essere causato da tre principali motivi:

- La sonda fluorescente può non essere espressa o non aver aderito in una specifica zona del batterio (esistono diversi tipi di sonde fluorescenti, le più comuni vengono prodotte direttamente dall'organismo geneticamente modificato come proteine, altre

vengono aggiunte nel terreno di coltura e si vanno a legare in particolari siti);

- Può essere presente una vescicola che oscura il segnale in una particolare zona;
- Potrebbe verificarsi il fenomeno di scissione binaria (riproduzione dei batteri) in cui non è chiaro se definire il batterio unico;

Alla luce di questi problemi mi è stato consigliato di considerare un batterio spezzato come due batteri distinti.

Dopo una prima ricerca di eventuali problemi fatta ad occhio nudo ho condotto un'analisi più attenta sui valori dell'immagine, in modo da poter scegliere il metodo migliore di segmentazione.

Dalla rappresentazione in 3D in figura 3.2 dei valori dell'immagine catturata con esposizione 12s ho potuto confermare come le immagini ad alta esposizione fossero affette da problemi di saturazione, ed è stato anche possibile notare un problema di vignetting accentuato soprattutto nella parte inferiore destra. Dall'immagine appare immediato il fatto che il background dell'immagine sia curvo.

Esistono vari modi per “appiattare” un'immagine affetta dal vignetting, ma nella maggior parte di questi è necessario acquisire una o più immagini prima di acquisire l'immagine su cui si fanno davvero le misure. Siccome il progetto è incentrato proprio sull'appesantire il meno possibile l'utilizzo del software da parte dell'utente finale ho deciso di non utilizzare nessuna di queste tecniche ma di risolvere il problema con la segmentazione.

3.1.2 Scelta del metodo di segmentazione

Come detto nella sezione 3.1.1 è necessario scegliere un metodo di segmentazione che risolva il problema del vignetting, quindi i metodi di sogliatura globale vengono esclusi, e siccome la segmentazione deve essere il più trasparente possibile per l'utente devono essere esclusi anche tutti i metodi di sogliatura supervisionati.

I test preliminari che ho fatto per scegliere l'algoritmo di sogliatura adatto riguardano principalmente quelli basati sulle statistiche sul vicinato: l'algoritmo di *Sauvola* e quello di risonanza.

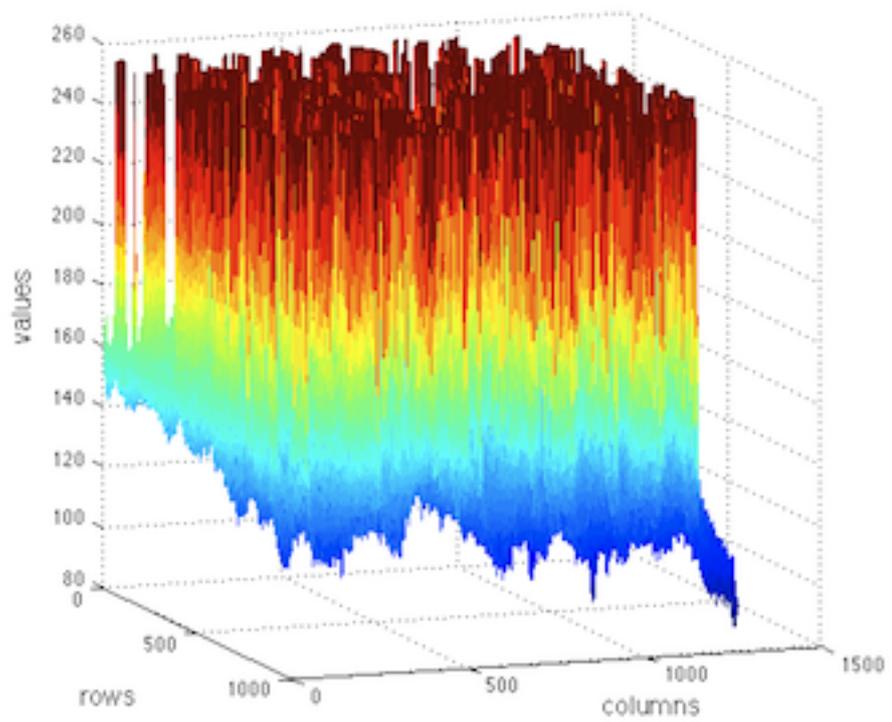


Figura 3.2: Rappresentazione in 3D dell'immagine intera con esposizione 12s

Degli algoritmi basati sulle statistiche del vicinato, quello che ha dato risultati migliori è certamente quello in cui si calcola la media dei vicini, spesso però questo porta a risultati non ottimi, quindi è necessario aggiungere una costante alla soglia calcolata. Quest'ultimo passaggio risulta troppo dipendente dall'immagine, e quindi necessita dell'intervento dell'utente.

Ciò che ha escluso l'utilizzo dell'algoritmo di risonanza è stato il gradiente presente intorno ai batteri, dove il loro bordo non è marcato nettamente e in questa situazione non si ha una buona risposta. Ulteriore punto a sfavore di questo algoritmo è la lentezza; siccome è definito con una struttura ricorsiva è molto pesante dal punto di vista computazionale.

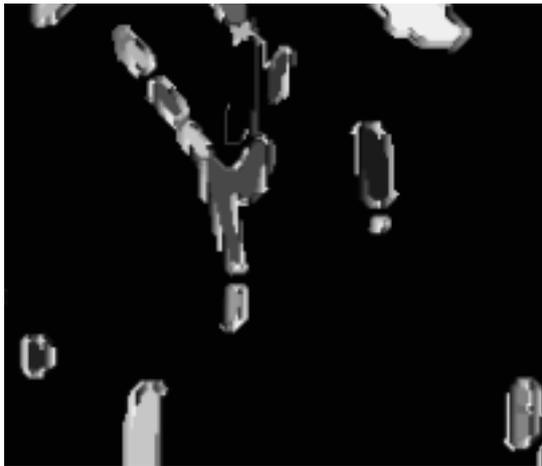


Figura 3.3: Particolare di segmentazione con algoritmo di risonanza

La scelta è ricaduta sull'algoritmo di *Sauvola*[14] che dà ottimi risultati e risolve anche il problema dello spostamento della soglia rispetto al valore di media calcolato. Lo spostamento viene calcolato in modo proporzionale alla deviazione standard dei vicini. Anche in questo algoritmo è presente un ulteriore parametro (i.e. k) che però è molto meno dipendente dal contenuto dell'immagine ma dipende solo dall'esposizione utilizzata.

Con gli algoritmi di sogliatura è difficile decidere dove inserire un bordo netto, così è stato necessario usare anche un *edge detector*: il *Canny Operator*.

Ottenuta la sogliatura e i bordi dei batteri, è necessario unire le due informazioni per produrre una segmentazione unica. Per ottenere il risultato è stato studiato un algoritmo euristico che prende spunto dall'operatore *hit-and-miss* ma con diverso metodo di applicazione. L'algoritmo usa come input l'immagine binaria ottenuta dall'*edge detector*, l'informazione sulla magnitudo del gradiente dell'immagine originale e l'informazione sulla direzione del gradiente; le operazioni che compie vengono fatte sul risultato della sogliatura. Questo metodo - a cui ho assegnato il nome di *Refiner* - ha lo scopo di rifinire il risultato della sogliatura che spesso assegna un'area maggiore di quella reale ai batteri (soprattutto ad alte esposizioni).

Il metodo utilizzato consiste nel porsi su ogni pixel trovato dall'*edge detector*, trovare la direzione del gradiente in quel punto e costruire uno *structuring element* che tramite operazione di *and* "scavi" nel verso opposto alla direzione trovata. La procedura di "scavamento" procede nella direzione opposta a quella del gradiente fino a quando non si incontra un altro bordo, non si incontra il background e la magnitudo del gradiente continua a diminuire (in quella direzione). Vengono usati 8 tipi di *structuring element* in base alla direzione trovata, di seguito due esempi:

$$\begin{bmatrix} 1 & 1 & 1 \\ 0 & 1 & 1 \\ 0 & 0 & 1 \end{bmatrix} \quad \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 0 & 0 & 0 \end{bmatrix}$$

a sinistra lo *structuring element* utilizzato con direzione NE (direzione compresa tra 22.5° e 67.5°) e a destra quello utilizzato con direzione N (direzione compresa tra 67.5° e 112.5°). Gli altri elementi possono essere trovati ruotando questi.

Siccome il *Canny operator* non garantisce di fornire un bordo chiuso intorno all'oggetto, può capitare che spesso due oggetti molto vicini rimangano collegati per mezzo di pochi pixel, portando poi ad una "etichettatura" (*labeling*) errata. Per questo motivo ho deciso di utilizzare un'operazione di *opening* (con *structuring element* 3×3 square)

subito dopo la rifinitura.

Terminata questa procedura il processo di segmentazione viene concluso con uno stage di *labeling* seguito da un filtro basato sull'area, dove tutti gli oggetti trovati al di sotto di una certa area vengono scartati.

3.1.3 Misurazioni

L'utilità di questo software è dovuta al fatto che permette di condurre misurazioni in radiometria e i problemi analizzati fino ad ora sono tutti finalizzati ad ottenere questo traguardo. Da una conversazione con il "committente" (i.e. il biologo) è emersa la necessità di poter scegliere i batteri su cui fare le misurazioni e anche di avere delle misurazioni globali su tutta la scena. Ulteriore funzionalità importante è quella di poter identificare quali batteri appartengono a certe misurazioni, il che si traduce nel poter fare una selezione inversa, ovvero scegliere i batteri a partire dalle loro misurazioni.

3.1.4 Parameter Tuning

La soluzione proposta, così come formulata finora, richiede una specifica conoscenza dell'elaborazione delle immagini da parte dell'utilizzatore, questo perché ogni algoritmo di segmentazione assistita possiede dei parametri che richiedono un apposito *set-up*. Per rendere il più trasparente possibile questo stage di elaborazione è stato necessario fare un'analisi dei parametri da utilizzare, in questo modo può essere fornita la possibilità di far scegliere all'utente se utilizzare i parametri prestabiliti (con valori di default) per utenti meno esperti oppure operare in modo avanzato, consentendo all'utente più esperto di modificare opportunamente i valori.

Per scegliere i parametri migliori ho condotto un'analisi sperimentale su dei campioni che mi portasse ad una regola generale. Come detto precedentemente i parametri scelti non saranno "*hard coded*" negli algoritmi ma saranno solo dei suggerimenti all'utente, il quale potrà cambiarli e regolarli a piacimento.

Gli algoritmi che necessitano di essere “regolati” sono l’algoritmo di *Sauvola* e il *Canny Operator* presentati nella sezione 2.4.

Set di immagini

Il test degli algoritmi e il parameter tuning è stato eseguito su due set di immagini prese da alcune culture di batteri sottoposte al GFP. Ogni set è composto da immagini rappresentanti la stessa scena acquisita con esposizioni diverse, il primo set riporta una scena con bassa densità di batteri mentre il secondo set riporta una scena con alta densità di batteri. I due set sono stati acquisiti rispettivamente a $\{1/1.5s, 1s, 1.5s, 3s, 8s, 12s, 15s\}$ e $\{1/1.5s, 1s, 1.5s, 2s, 3s, 4s, 6s, 8s\}$ di esposizione.

Filtro gaussiano

Prima del passo di *segmentazione* e quello di *edge detection* è stato usato un filtro gaussiano per rendere l’immagine lievemente più continua nei valori, e quindi eliminare eventuali rumori. Il raggio¹ di questo filtro è stato impostato a 2 di default poiché aumentandone la dimensione il risultato non migliorava.

Sauvola

Questo algoritmo ha due parametri principali che sono la grandezza della finestra con cui si scansiona l’immagine (*window*) e il parametro k .

Il primo parametro influenza la valutazione della soglia, scegliendo se considerare solo i pixel più vicini a quello analizzato oppure se considerare anche quelli più lontani. Una buona pratica è scegliere quest’ultimo in base alla dimensione degli oggetti da segmentare. Il grosso problema della stima di questo parametro - nell’applicazione corrente - è che all’aumentare dell’esposizione con cui si acquisisce l’immagine la dimensione dei batteri cresce. Questo problema è dovuto al fatto che i batteri irradiati sono fluorescenti, quindi producono

¹D’ora in poi userò la parola raggio riferendomi ad una finestra in modo improprio, con raggio verrà intesa la metà della lunghezza del lato della finestra meno uno (raggio = (lato della finestra - 1) / 2)

intorno alla propria silhouette un gradiente di luminescenza.

Un altro problema che caratterizza la scelta di questo parametro è la cardinalità dei batteri all'interno dell'acquisizione: se il numero di batteri è elevato è molto probabile che ci siano sovrapposizioni. Una sovrapposizione non può essere riconosciuta e la possibilità di riconoscerla diminuisce all'aumentare dell'esposizione. Questo problema porta ad un'inevitabile aumento del parametro *window* all'aumentare della densità dei batteri e nonostante l'aumento del parametro sarà difficile segmentare separatamente i batteri addensati.

Quando si processa un oggetto molto vasto senza un valore di luminescenza uniforme sulla sua superficie, se si utilizza una finestra piccola rispetto a quella necessaria, è possibile che si abbia come risultato una segmentazione con delle parti interne all'oggetto mancanti.

Alla luce di questi problemi si è potuto evincere che la grandezza della finestra dovesse essere determinata con una funzione dipendente dall'esposizione e dalla densità dei batteri.

La determinazione della densità di batteri in un'immagine non è un "task" semplice da compiere. Per fare ciò è necessario eseguire una segmentazione "grossolana" iniziale e poi valutare la porzione di immagine occupata dagli oggetti. Questa segmentazione è stata effettuata tramite l'algoritmo di *Sawvola* con dei parametri di base che vedremo in seguito.

In prima battuta ho deciso di utilizzare un algoritmo semplice che portasse ad un rapido calcolo della percentuale di immagine occupata dagli oggetti: ho calcolato il numero di pixel segmentati sul totale dei pixel. Questa soluzione porta con sé un errore dovuto al fatto che l'area dei batteri cresce all'aumentare dell'esposizione, quindi due immagini della stessa scena ad esposizioni diverse portano ad un risultato completamente diverso. Normalizzare il risultato con l'esposizione non migliora la situazione.

Il secondo metodo che ho applicato è stato il conteggio delle istanze di batteri nell'immagine, dopo una prima fase di segmentazione ho utilizzato una fase di *labeling*. Anche questo metodo è risultato inutilizzabile poiché ad esposizioni basse i batteri sono ben distinti e quindi facilmente contabili, aumentando l'esposizione formano dei cluster che non è possibile scindere, ciò produce nella valutazione una diminuzio-

ne delle istanze all'aumentare dell'esposizione.

La penultima tecnica che ho adottato per risolvere questa valutazione è scaturita dall'osservazione che il *Canny Operator* è molto meno sensibile all'esposizione rispetto alla segmentazione con *Sauvola*. Modificando il metodo proposto in [12], ho sovrapposto una griglia all'immagine e ho contato tutte le intersezioni degli edge (calcolati con l'edge detector) con la griglia, in teoria questa valutazione avrebbe dovuto darmi un'approssimazione del numero di istanze. Anche questa tecnica risulta dare approssimazioni errate, probabilmente dovute a edge collineari alla griglia.

In ultima analisi ho adottato la tecnica suggerita in [12] che consiste nel segmentare l'immagine originale per averne una rappresentazione binaria, data la maschera si sovrappone una griglia e si controlla il valore di quest'ultima in corrispondenza degli incroci della griglia. La somma dei valori con riscontro positivo - normalizzato sulla totalità degli incroci - fornisce la densità. Questa tecnica corrisponde ad un campionamento equispaziato sull'immagine.

Il risultato di questa valutazione rimane comunque dipendente dall'esposizione ma dà approssimazioni migliori.

Sottoponendo il set di immagini ad una griglia di 2500 punti (50x50), quelle con densità minore vengono valutate con una densità che varia nel range 1% - 4% mentre quelle con densità maggiore vengono valutate con densità nel range 20% - 40%.

Ottenuto un valore indicativo per la densità abbiamo a disposizione tutte le misure necessarie per computare la grandezza della finestra. Come detto in precedenza è stato seguito un metodo sperimentale, che ha portato ai risultati in tabella 3.1. Si può notare dai risultati che per una bassa esposizione ($< 3s$) può essere utilizzata in entrambi i casi una finestra con raggio 5 (ovvero una finestra 11x11), all'aumentare dell'esposizione è necessario definire una funzione per il calcolo della grandezza della finestra. È doveroso ricordare che la variazione di 1 o 2 pixel sulla dimensione del raggio della finestra non porta ad un sostanziale cambiamento sul risultato.

Calcolandone un'approssimazione, la distribuzione ottenuta dai test sulle immagini a bassa densità verrà usata come finestra di base sia per il *pre-processing* (sogliatura iniziale per determinare la densità dell'immagine), sia per la valutazione dei parametri di default. Nella

Esposizione	Set bassa densità	Set alta densità
1/1.5s	5	5
1s	5	5
1.5s	5	5
2s	-	5
3s	5	7
4s	-	8
6s	-	12
8s	6	24
12s	10	-
15s	13	-

Tabella 3.1: Valutazione parametro window

figura 3.4 un grafico della distribuzione sperimentale (linea continua) e dell'approssimazione (linea tratteggiata).

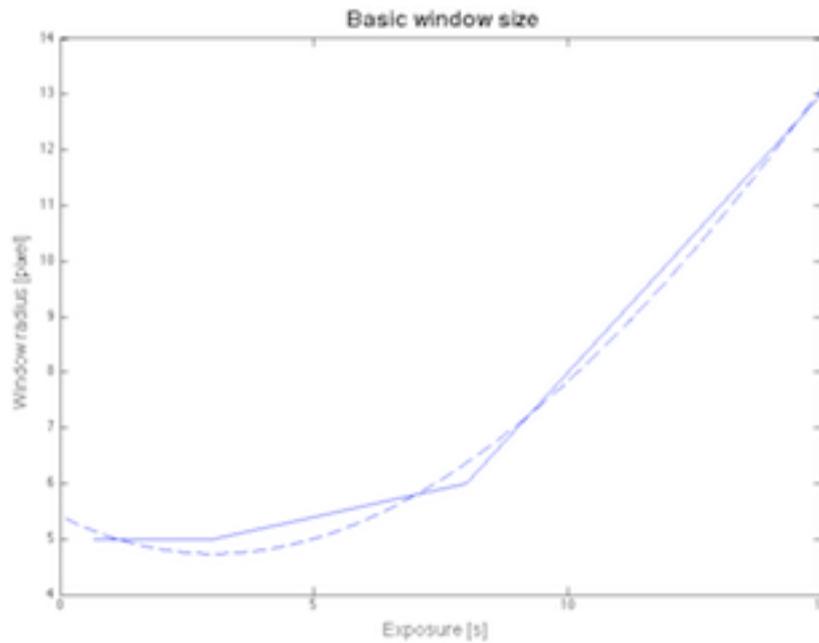


Figura 3.4: Finestra di base utilizzata per la segmentazione

L'approssimazione è stata calcolata con un *data fitting* utilizzando una polinomiale di 3° grado. Questa ha dato origine alla funzione:

$$y(x) = -0.001063x^3 + 0.08082x^2 - 0.4611x + 5.423$$

Il grafico mostra che se si ha un'esposizione minore di 10s si può incorrere in un raggio della finestra minore di 4 e se si ha un'esposizione vicino a 0 si può avere una finestra maggiore di 5, questo può essere controllato via software impostando come limite minore della finestra il valore 5 e impostando che, con un'esposizione $< 3s$, la finestra è sempre di raggio 5. Ottenuta la finestra di base, con lo stesso metodo, ricavo una funzione che esprime il rapporto dei raggi delle finestre ottenuti con il metodo sperimentale (vengono considerate solo le esposizioni $\{1/1.5s, 1s, 1.5s, 3s, 8s\}$ poiché le uniche presenti in entrambi i set). In altre parole, viene cercata una funzione che data l'esposizione ricavi il rapporto tra il raggio della finestra di base e quello della finestra di cui si necessiterebbe. Di seguito viene mostrata la distribuzione del rapporto dei raggi (linea continua) e l'approssimazione (linea tratteggiata):

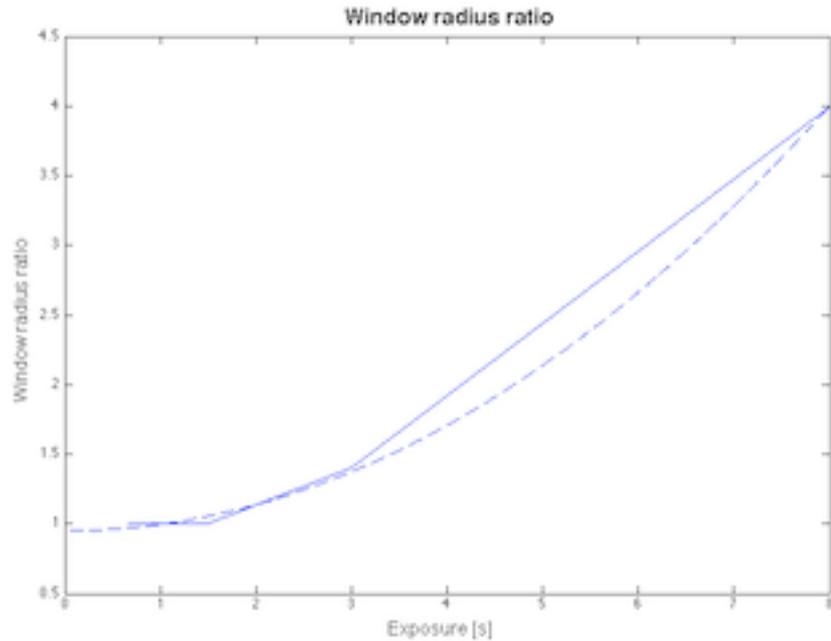


Figura 3.5: Rapporto tra i raggi delle finestre

L'approssimazione con una polinomiale di 2° grado dà origine alla funzione:

$$y(x) = 0.04804x^2 - 0.003069x + 0.9518$$

Trovata la dipendenza del rapporto dall'esposizione è necessario modulare il “peso” di quest'ultimo in base alla densità dell'immagine, supponendo la distribuzione del rapporto lineare con la crescita della densità si ha:

$$window = w_{base} \left(1 + (ratio - 1) \cdot \frac{density - 0.01}{0.39} \right)$$

dove w_{base} rappresenta il raggio della finestra di base, $ratio$ rappresenta il rapporto calcolato precedentemente e $density$ rappresenta la densità calcolata con il metodo della griglia. Al termine della computazione si avrà chiaramente un numero non intero, questo verrà arrotondato all'intero più vicino poiché non è possibile utilizzare finestre continue ma solo discrete.

Il parametro k influenza la valutazione determinando quanto può essere abbassata la soglia calcolata con la media dei vicini in base alla loro deviazione standard. Questo fornisce la possibilità di rendere più sensibile o meno l'algoritmo alle piccole variazioni di luminosità nell'immagine. Eseguendo dei test sul set di immagini è risultato:

Esposizione	Set bassa densità	Set alta densità
1/1.5s	0.01	0.01
1s	0.01	0.01
1.5s	0.01	0.01
2s	-	0.02
3s	0.02	0.02
4s	-	0.02
6s	-	0.02
8s	0.05	0.05
12s	0.05	-
15s	0.1	-

Tabella 3.2: Valutazione parametro k

Come si può vedere, questo parametro non è influenzato dal numero di batteri presenti nell'immagine ma esclusivamente dall'esposizione dell'acquisizione stessa. In questo caso la determinazione del parametro è molto più semplice e può essere ricavata facilmente da un'approssimazione di 2° grado, come si può vedere in figura:

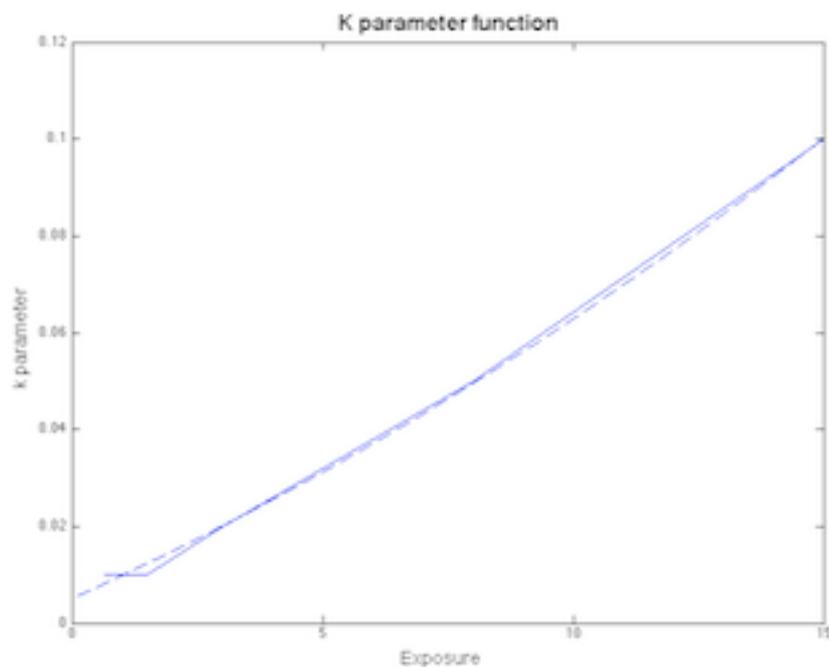


Figura 3.6: Distribuzione del parametro k

La linea tratteggiata che approssima la distribuzione del parametro k rappresenta la funzione:

$$y(x) = 0.000112x^2 + 0.004657x + 0.005061$$

Nonostante nell'immagine la funzione vada anche al di sotto del valore 0.01, nel software verrà tenuto come minimo il valore 0.01.

Canny Operator

L'algoritmo presenta due parametri: la soglia maggiore e la soglia minore. La soglia minore ho deciso di fissarla a $2/5$ della soglia maggiore.

La soglia maggiore è responsabile della sensibilità dell'algoritmo, se il valore si abbassa basterà un minore "dislivello" di luminosità nell'immagine per formare un *edge*. Come per i parametri precedenti ho eseguito dei test sui set di immagini, questi mi hanno portato a notare che l'algoritmo è insensibile alla densità dei batteri e quindi è stato possibile definire il valore in funzione dell'esposizione in modo lineare:

$$\begin{cases} 30 & exp \leq 8s \\ 30 + (exp - 8) \cdot 2.5 & exp > 8s \end{cases}$$

3.1.5 Macchina a stati

Impostando un modello "guidato" di interazione con l'utente (*wizard*) ne scaturisce la macchina a stati in figura 3.7. Come si può ben vedere l'applicazione di un *image processor* non porta allo stato successivo, in questo modo l'utente può verificare il risultato e eventualmente correggerlo. Nello stato di *Init* vengono calcolati i parametri di default e può essere scelto il canale dell'immagine su cui si vogliono fare le misurazioni. Negli stati di *Measuring* sarà possibile fare una selezione dei batteri e dallo stato finale in poi sarà possibile recuperare questa misura.

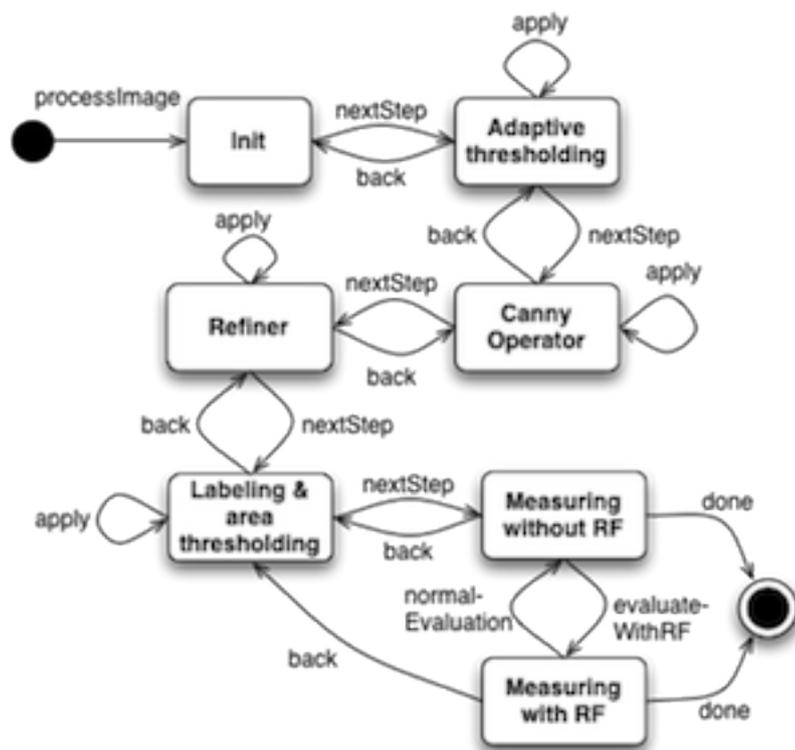


Figura 3.7: Macchina a stati del motore di elaborazione

3.2 Interfaccia grafica

In questa sezione viene fatta un'analisi del problema dal punto di vista dell'interfaccia grafica, vengono spiegate tutte le scelte progettuali legate al layout scelto e alla fruibilità dei dati.

3.2.1 Layout

Facendo una banale osservazione sul formato di un'immagine (rapporto tra le dimensioni) e sul formato di un monitor, si può facilmente notare che entrambi si estendono in maggior misura nella direzione orizzontale rispetto a quella verticale (la valutazione è valida per i monitor più recenti e immagini generiche in formato landscape). Questo mi ha portato a ideare un'interfaccia in cui i comandi siano posti nella parte laterale della finestra, in questo modo l'immagine su cui si lavora ha a disposizione un'area più ampia per essere visualizzata (questo verrà utile quando si agirà direttamente sull'immagine).

L'area dei comandi verrà divisa in due parti sia logiche che fisiche, la parte superiore conterrà i comandi per la segmentazione e lo stato attuale, mentre la parte inferiore conterrà i risultati delle misurazioni. Ho scelto di mantenere queste due sezioni distinte per mantenere una spartizione ben chiara anche nella mente dell'utilizzatore. A mio parere molte interfacce grafiche spesso nascondono parti - per sfruttare al meglio lo spazio - rendendone poi difficile il ritrovamento per un utente inesperto.

La dimensione dell'area dei comandi verrà resa fissa in modo che quando la finestra verrà espansa non sottrarrà spazio alla visualizzazione dell'immagine. In figura 3.8 viene mostrato lo schema di layout generale della finestra visualizzata:

Control area

Siccome la macchina a stati prevede che vengano eseguite elaborazioni in successione per raggiungere la segmentazione, ho deciso di comporre questa fase come un *wizard*, tecnica comune anche agli utenti meno esperti. Durante questa fase nella *control area* si susseguono i parametri che possono essere impostati ad ogni elaborazione.

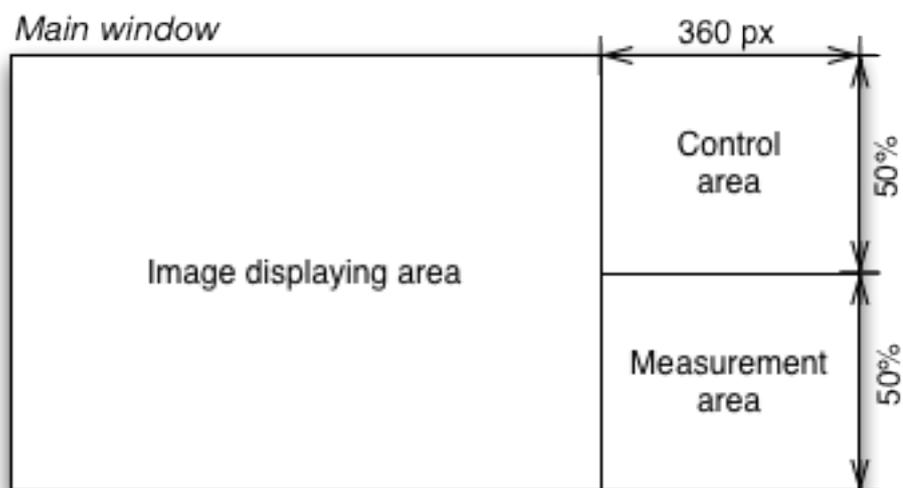


Figura 3.8: Layout della finestra principale

Measurement area

In questa zona verranno visualizzate le misure ottenute dall'immagine, siccome è necessario visualizzare sia le misure globali che le misure locali scelte dall'utente, ho pensato di intercambiare le visualizzazioni con un pannello a schede. Con questa tecnica si riesce ad utilizzare in entrambi i casi la completa area a disposizione. Per sfruttarla al meglio, ho deciso di visualizzare in alto le varie misure e sotto le anteprime dei grafici, spesso nei grafici è molto più interessante l'andamento che le singole misure.

L'area occupata dalle misure va fornita di una barra di scorrimento per permettere la fruizione dei risultati anche con la finestra di dimensioni più piccole.

Image displaying area

In questa area verrà mostrata sempre l'acquisizione originale in tutti i suoi canali, ad ogni step di segmentazione gli verrà sovrapposto in trasparenza il risultato dell'elaborazione stessa. Questo metodo garantisce un modo facile per l'utente per fare un confronto tra il risultato ottenuto e ciò che si aspettava.

Nella fase di misurazione quest'area dovrà anche mostrare i *Minimum Enclosing Rectangle* dei batteri selezionati dall'utente.

3.2.2 Interazione con l'utente

Una parte molto importante per l'interfaccia grafica è cercare di fare intuire anche a un utente poco esperto ciò che sta succedendo o ciò che potrebbe fare in quel momento con il mouse.

Durante il “*wizard*” di segmentazione si hanno dei momenti in cui l'interfaccia grafica deve comunicare all'utente che il sistema sta lavorando, in questo momento è necessario anche disabilitare la possibilità di inserire valori o di premere pulsanti per evitare che l'utente vada ad interferire con l'elaborazione. Questa strategia viene utilizzata quando il tempo di computazione è piuttosto elevato, per i tempi più brevi basta semplicemente modificare il puntatore del mouse inserendo quello della modalità di attesa.

Particolare accortezza viene data proprio alla scelta dei puntatori che andranno cambiati in base all'operazione possibile da parte dell'utente, nella fase di misurazione il mouse andrà convertito nel tipico puntatore “con il dito indice alzato” per far capire all'utente quali batteri fanno parte della segmentazione (e quindi possono essere selezionati) e quali sono stati esclusi (per esempio con l'*area filtering*).

3.2.3 Interazione dell'utente

La politica adottata per l'impostazione delle operazioni a disposizione dell'utente si basa su un semplice principio: per compiere un'azione esiste solo un modo. Questo è uno dei motivi che mi ha portato a non inserire una barra degli strumenti nell'interfaccia, poiché credo che la possibilità di fare la stessa cosa in molteplici modi diversi insinui dei dubbi nell'utilizzatore poco esperto.

Di seguito verranno descritte le interazioni più di rilievo che sono sorte analizzando i requisiti del problema posto.

Selezione diretta

Dopo aver concluso la fase di segmentazione l'utente deve aver la possibilità di scegliere uno o più batteri su cui recuperare una misura

aggregata. Questa operazione deve essere possibile solo quando si è nella parte delle misurazioni locali.

Per una massima flessibilità, si deve dare a disposizione come strumento di selezione il click singolo oppure la possibilità di selezione di un'area rettangolare; per aumentare ancora di più la libertà dell'utilizzatore è bene mettere a disposizione anche la possibilità di fare scelte multiple utilizzando un tasto della tastiera (i.e. tasto shift) oltre al mouse. Un ulteriore accorgimento è la possibilità di deselezionare un batterio già selezionato sempre per mezzo dello stesso tasto, quindi togliere dall'insieme degli elementi selezionati un elemento se è già presente. Questa operazione verrà fatta anche con la selezione rettangolare, quindi se tutti i batteri appena selezionati saranno già presenti nell'insieme di quelli selezionati precedentemente, questi verranno tolti.

In figura 3.10 si può vedere la macchina a stati studiata per il modello di selezione, le azioni che generano il movimento da uno stato all'altro sono chiaramente quelle compiute con il mouse dall'utente. Gli stati identificano rispettivamente:

Selection stato in cui il tasto del mouse è stato premuto ma non ancora rilasciato,

Rectangle drag stato in cui il tasto del mouse è stato premuto e si sta trascinando il mouse,

Single selection stato in cui il tasto del mouse è stato rilasciato dopo esser stato premuto, senza trascinamento del mouse,

Rectangle selection stato in cui il tasto del mouse è stato rilasciato dopo che il mouse è stato trascinato,

Multiple selection stato in cui il tasto del mouse è stato premuto, mentre il tasto sulla tastiera era premuto, e non ancora rilasciato,

Multi Rectangle drag equivalente di *Rectangle drag* ma dopo che il tasto del mouse è stato premuto insieme al tasto della tastiera,

Add Rectangle stato in cui si rilascia il tasto del mouse dopo aver trascinato il mouse

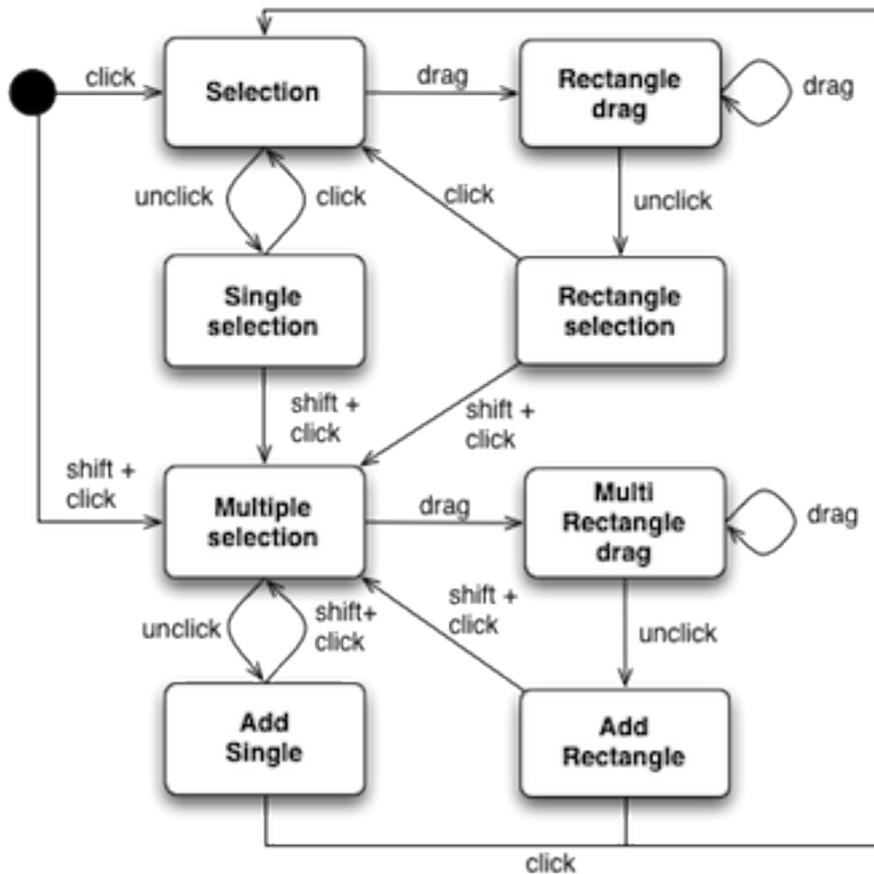


Figura 3.9: Macchina a stati per la selezione dei batteri

Add Single stato in cui si rilascia il tasto del mouse dopo averlo premuto contemporaneamente al tasto della tastiera.

Selezione inversa

Con selezione inversa si intende quella selezione che viene eseguita sui dati stampati sul grafico per vedere quali batteri corrispondono ad essi. La selezione inversa può essere molto utile per identificare gli *outliers* oppure per selezionare solo un sottoinsieme dei risultati che soddisfano criteri di selezione in un intervallo specifico di valori. Per compiere questa azione, deve esser possibile selezionare una parte

del grafico mediante il ridimensionamento di un rettangolo con bordi attivi, è inoltre necessario mostrare le misure a cui verrà fatta la selezione (quindi i valori a cui sono posti i bordi del rettangolo) in modo che l'utente possa scegliere con consapevolezza che parte del grafico evidenziare.

I batteri selezionati verranno identificati nello stesso spazio in cui vengono identificati quando si fa la selezione diretta.

Per compiere questa azione le anteprime dei grafici poste nella *Measurement area* sono troppo piccole e quindi la selezione sarebbe poco precisa. Per ovviare a ciò è necessario ingrandire le anteprime dei grafici quando vengono cliccati, dando così l'opportunità all'utente di analizzare questi ultimi anche con un dettaglio maggiore.

L'operazione di selezione inversa non è adatta agli istogrammi delle intensità poiché come risultato si avrebbero solo pixel evidenziati sparsi per l'immagine e non darebbero alcun contributo informativo ulteriore all'utilizzatore.

3.2.4 Grafici

Dal colloquio con il biologo è emerso che i grafici più importanti sono quelli dell'intensità luminosa dei batteri e quelli della loro area sulla luminosità media, così è necessario creare due tipi di grafici: un'istogramma e un grafico a dispersione. È stato scelto come tipo di grafico quello a dispersione perché con la strategia di filtraggio che sarà implementata estendendo il rettangolo di selezione in orizzontale si può fare una selezione *a banda* sulle ascisse, estendendolo invece in verticale si potrà effettuare una selezione a banda sulle ordinate.

Entrambi i grafici devono avere la stessa caratteristica di zoom ma solo il secondo tipo deve essere filtrabile (avere la possibilità di eseguire una selezione inversa).

Per aumentare la fruibilità dei grafici verrà aggiunta la lettura dei valori al passaggio del mouse, quando il mouse è sopra un dato stampato sul grafico è necessario visualizzare il suo esatto valore.

3.2.5 Stile

Per quanto lo stile grafico sia una caratteristica piuttosto secondaria nelle applicazioni scientifiche comunque riveste un ruolo importante. La produzione di un'applicazione con un'estetica molto basilare dà all'utente inesperto una sensazione di poca cura nello sviluppo.

I colori utilizzati non devono essere casuali, il background dell'immagine è opportuno sceglierlo con un colore che contrasti l'immagine stessa in modo da evidenziarne i contorni, siccome il background potrebbe essere di qualsiasi colore è buona norma utilizzare una *texture* oppure un gradiente.

Ho deciso di basare la scelta dei colori sulla teoria dei colori complementari [11], questi colori vengono praticamente identificati dall'occhio come quelli più contrastanti possibili, ovvero quelli di cui l'occhio riesce a percepire meglio la differenza. Il background verrà colorato con un grigio scuro neutro, credo che questo porti ad un lavoro più rilassato per gli occhi dell'utilizzatore rispetto ad un bianco o ad un colore chiaro, le scritte saranno lievemente più chiare per creare del contrasto. I grafici utilizzeranno una tonalità viola abbastanza chiara per risaltare sul background scuro, su questi verrà poi utilizzato un filtro giallo scuro tendente all'arancione, poiché opposto al viola nella teoria dei colori complementari. Per quanto riguarda la visualizzazione dei *Minimum Enclosing Rectangle* che dovranno essere disegnati sull'immagine originale, ho scelto il rosso, colore complementare del verde sprigionato dal *GFP*.



Figura 3.10: Colori complementari: sono detti complementari i colori opposti nel cerchio (ovvero con tinta opposta).

Capitolo 4

Implementazione

In questo capitolo viene analizzata la traduzione dell'analisi del problema (fatta nel capitolo precedente) nel software vero e proprio. Qui vengono affrontate tutte le tematiche relative all'implementazione, perciò dipendenti dalla tecnologia utilizzata.

L'applicazione deve essere sviluppata come *plug-in* per un sistema già funzionante, quindi non deve aver alcun *main* ma deve essere distribuita come libreria (*assembly* in ambito *.NET*) e non come eseguibile. Il sistema principale è stato sviluppato in linguaggio *C#* poiché ritenuto più adatto per un interfacciamento con i driver della fotocamera di acquisizione, quindi, per un accoppiamento perfetto è necessario che anche il *plug-in* venga sviluppato con tecnologia *.NET*. Per quanto riguarda la tecnologia utilizzata per l'interfaccia grafica avevo a disposizione due scelte: la prima era quella di utilizzare il classico ambiente *Windows Form*, la seconda era quella di utilizzare il più recente xml-based *Windows Presentation Foundation WPF*. La volontà di produrre un'interfaccia grafica moderna mi ha portato subito alla scelta del *WPF*, però questa non è stata una scelta esclusivamente stilistica, questo nuovo strumento permette di definire delle interfacce grafiche utilizzando un linguaggio xml-based (*xaml*) rendendolo molto simile a *HTML + CSS* usati per la definizione di layout per interfacce web. Ogni controllo *WPF* che viene definito è formato da due file che solo insieme definiscono completamente il componente, uno è la parte di definizione visuale (*.xaml*) e uno la parte di definizione computazionale (*.xaml.cs*); spesso però questa distinzione non è così

netta.

4.1 Struttura della soluzione

La soluzione che ho creato è suddivisa logicamente (e fisicamente) in tre *namespace* di primo livello separati (che danno origine a tre progetti diversi nell'ambiente di sviluppo):

MainApplication è il namespace che simula il programma padre che invocherà il *plug-in*, viene utilizzato solo in fase di test e non verrà ridistribuito, la sua funzione principale è quella di *image provider*;

Measurement è il namespace che diventerà poi la libreria da distribuire e da utilizzare come *plug-in*, contiene al suo interno tutta la logica;

ImagingChart è il namespace in cui vengono definiti i grafici per stampare a video i dati dell'*image analysis*;

i tre “pacchetti” hanno una relazione di utilizzo, **Measurement** utilizza **ImagingChart** per funzionare e **MainApplication** utilizza **Measurement**. Di seguito verrà analizzata la struttura dei singoli namespace.

4.1.1 MainApplication

Questo namespace è molto semplice poiché rappresenta solo il *bootstrap* del sistema in fase di test, contiene al suo interno solo la definizione di una semplice *Form* con cui si può recuperare un'immagine dal file-system e di cui se ne può inserire l'esposizione. Recuperate queste due informazioni si passa il controllo al *plug-in*.

4.1.2 ImagingChart

Questo namespace contiene al suo interno la definizione dei grafici e tutto ciò che necessita per il loro utilizzo, il namespace può essere

compreso in un *assembly* e distribuito come libreria di *User Control*¹ per un eventuale utilizzo da parte di altri software.

In figura 4.1 è possibile vedere la struttura interna del namespace.

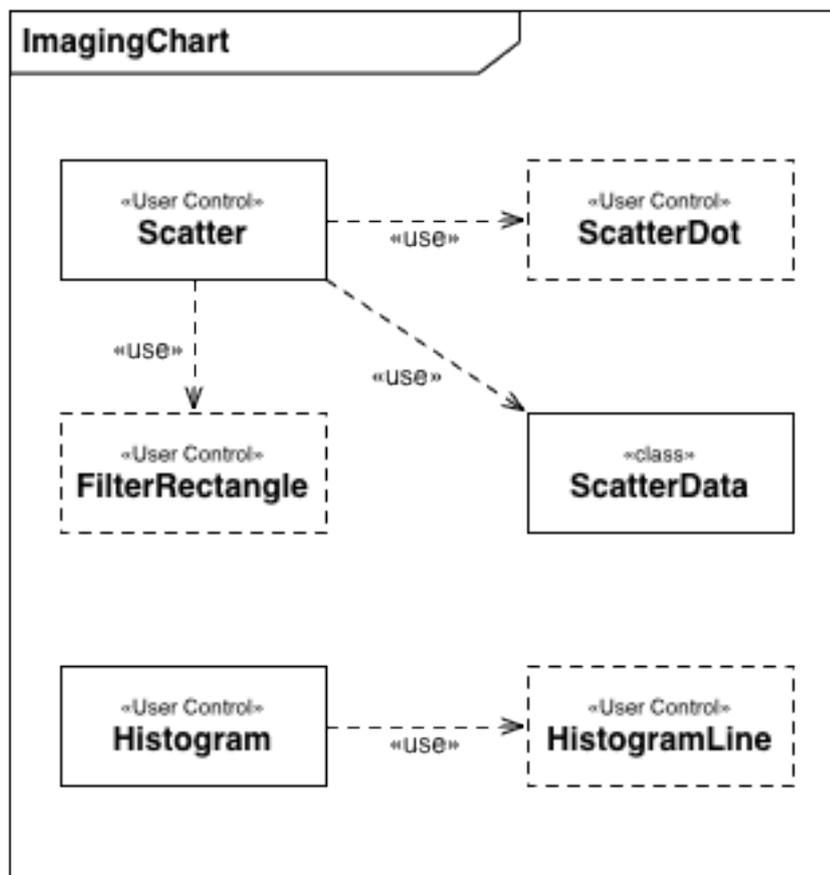


Figura 4.1: Struttura interna del namespace `ImagingChart`. Classi pubbliche con la linea continua e classi private con la linea tratteggiata.

`Scatter` rappresenta il grafico a dispersione e include al suo interno anche tutta la logica per fare in modo che vengano visualizzati i dati in una label quando il mouse viene sovrapposto ad essi. Ognuno di questi

¹L'*User Control* in *WPF* è un elemento dell'interfaccia grafica definito dal progettista per mezzo di aggregazione di altri elementi già esistenti nella libreria di base.

dati è rappresentato visivamente da un oggetto di tipo `ScatterDot` e dal punto di vista del modello da un oggetto di tipo `ScatterData`, è stata fatta questa distinzione in modo che non sia necessario per un'utente esterno conoscere l'esistenza dell'implementazione visiva ma può semplicemente passare i dati al grafico attraverso l'implementazione del modello (i.e. `ScatterData`). Questo tipo di grafico ha la possibilità di eseguire una selezione inversa attraverso un rettangolo di selezione (i.e. `FilterRectangle`) di cui si può controllare posizione e dimensione. Questa caratteristica può essere anche disabilitata se non si vuole che il grafico sia filtrabile.

`Histogram` rappresenta un grafico della tipologia "istogramma" e viene utilizzato in questo ambito per mostrare l'istogramma delle intensità dei pixel, ogni barra dell'istogramma è di tipo `HistogramLine`. Anche questo tipo di istogramma implementa al suo interno la funzionalità di visualizzazione dei valori al passaggio del mouse.

4.1.3 Measurement

Questo namespace contiene tutta la logica dell'applicazione, dovrà poi essere distribuito come *assembly* per poter esser utilizzato dall'environment principale come *plug-in*. La struttura interna è suddivisa ulteriormente in namespace di secondo livello che rispecchiano un'implementazione con pattern *MVC* (Model View Control). L'unione delle figure 4.2 e 4.3 produce il diagramma completo della struttura intera.

Measurement.Model

In questo namespace sono contenute tutte le classi che si riferiscono al modello studiato, solo le interfacce relative alle misure e alla rappresentazione dell'immagine sono state rese pubbliche, essendo le uniche necessarie all'esterno.

Innanzitutto è stato necessario definire una rappresentazione interna delle immagini per poterci lavorare, da questo punto di vista le librerie di base che offre *C#* sono piuttosto limitate. Il linguaggio fornisce ottimi strumenti per la visualizzazione delle immagini o per il disegno ma non per l'elaborazione. L'interfaccia `IImageMatrix` e la relativa

classe `ColorImage` rappresentano un'immagine sotto forma di matrice di `double`, in modo che possa essere il più generale possibile e utilizzabile in tutti i contesti applicativi.

Le misurazioni sono divise in due categorie:

SingleMeasure Sono delle misure che riguardano un solo batterio, contengono all'interno tutte le misure che lo riguardano, quelle statistiche, i valori da cui è formato, il bounding box, ...;

MultiMeasure Rappresenta una misura aggregata di più batteri, le misurazioni statistiche di questa misura sono fatte sulle statistiche delle singole misure di ogni batterio;

Per rendere più rapida l'analisi dell'immagine è necessario crearne una rappresentazione alternativa. In questa rappresentazione vengono suddivise le parti interessanti dell'immagine originale in modo che possano essere raggiunte più velocemente per mezzo di un identificativo. Con questo metodo viene creata una struttura (i.e. `IImageStructure`) da cui è più facile calcolare statistiche sugli oggetti segmentati. Ogni oggetto segmentato a sua volta ha una sua rappresentazione interna che ne definisce sia i valori che le misure (i.e. `IObjectDetail`).

Fino ad ora sono state definite le parti del modello che riguardano l'*image analysis*, per quanto riguarda l'*image processing* sono stati creati due namespace: `Utils` e `Processors`. Nel primo namespace vengono definite tutte le operazioni utili quando si lavora con le immagini, come per esempio la convoluzione, questa classe ha solo membri statici, quindi non può essere istanziata ma offre solo degli strumenti. Il secondo namespace presenta tutti gli algoritmi che vengono utilizzati per elaborare l'immagine, implementano tutti la stessa interfaccia `IImageProcessor` poiché hanno tutti in comune dei parametri di ingresso e l'applicazione su un'immagine (siccome le immagini sono tutte codificate come `IImageMatrix` è necessario fare attenzione al tipo di immagine che verrà sottoposta all'algoritmo, se binaria, greyscale o double).

Measurement.Control

In questo namespace è presente la logica di controllo dell'applicazione, quella che fa da accoppiamento tra l'interfaccia grafica e il modello.

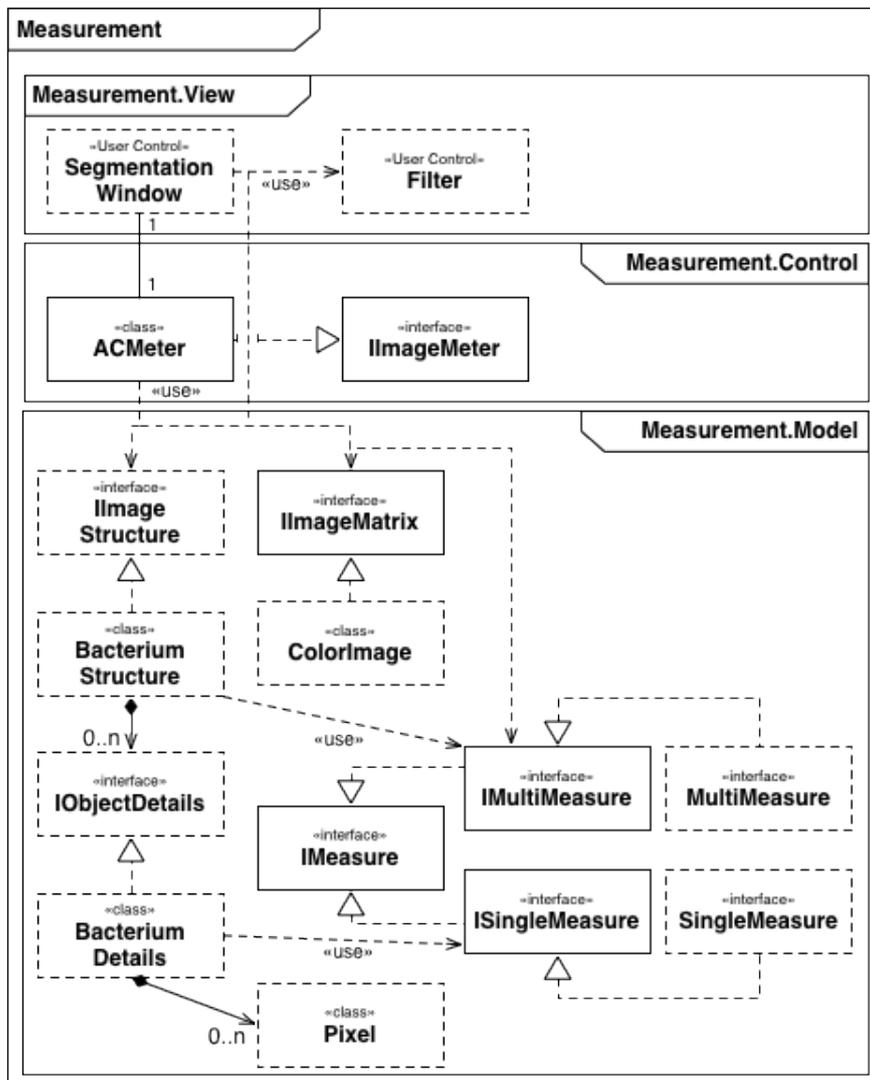


Figura 4.2: Struttura interna del namespace **Measurement**, sono rappresentati solo i namespace di secondo livello, quelli di terzo livello sono mostrati in figura 4.3. Classi pubbliche con la linea continua e classi private con la linea tratteggiata.

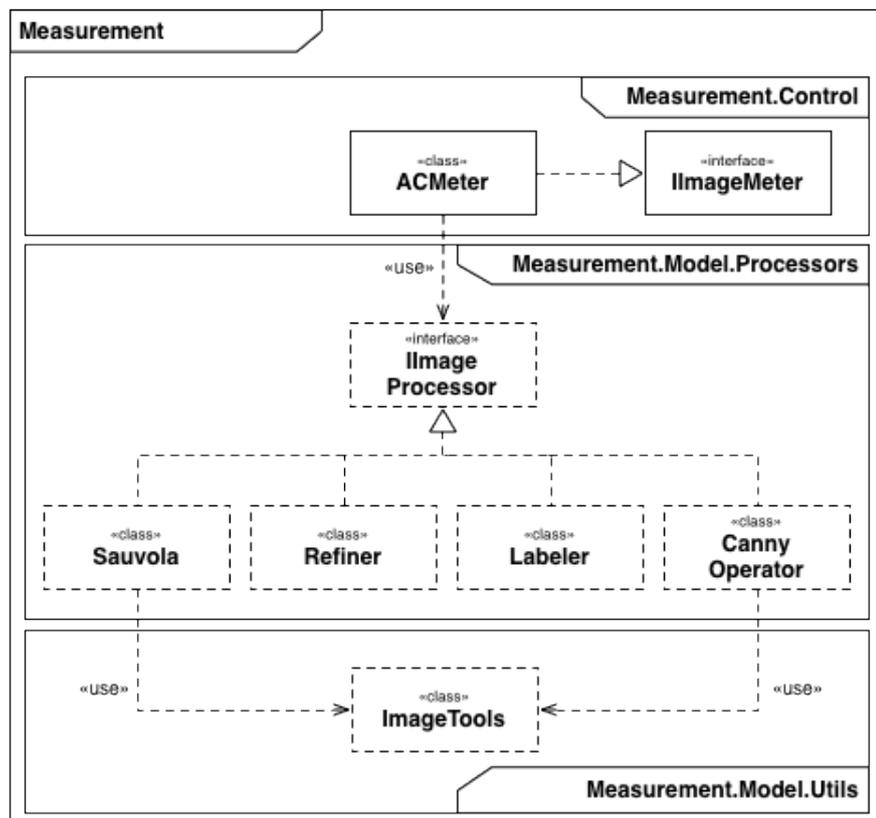


Figura 4.3: Struttura interna del namespace `Measurement`, namespace di terzo livello. Classi pubbliche con la linea continua e classi private con la linea tratteggiata.

In questa sede viene implementata la macchina a stati in figura 3.7. La classe `ACMeter` che implementa la classe `IImageMeter` è l'unica classe che ha contatto con il programma principale, identificato nella fase di test con `MainApplication`. Dall'interfaccia che questa classe offre è possibile impostare l'immagine da analizzare, far procedere la macchina a stati e recuperare i risultati. Ogni elaborazione viene fatta su un nuovo *thread*, garantendo così la continuazione del *thread* dell'interfaccia grafica, questo evita all'utente il fastidioso bloccarsi di quest'ultima. Data la natura *multithread* e asincrona del controllo è stato necessario anche impedire l'esecuzione di altri comandi durante un'elaborazione.

Questa classe si occupa anche di formare la struttura `IImageStructure`, richiedergli le misure e - in caso - di rivalutare le misurazioni con la funzione di risposta.

Measurement.View

Questo namespace è composto da due controlli che derivano direttamente da `Window` e rispecchiano le due finestre principali che vengono utilizzate nell'applicazione.

`Filter` è la finestra in cui vengono mostrati gli ingrandimenti dei grafici, dispone anche di una barra di controllo con dei pulsanti per abilitare o disabilitare la modalità di filtrazione e per eventualmente filtrare i dati. Questa finestra ha una relazione di conoscenza reciproca con l'anteprima del grafico in modo che possa essere aggiornata nel caso i dati cambiassero. Inoltre, provvede a comunicare all'anteprima la propria chiusura per evitare che vengano aggiornati dati inutilmente.

`SegmentationWindow` invece è la finestra principale del programma, segue esattamente il layout studiato nella sezione 3.8.

Nella parte che era stata identificata come *Image displaying area* è stato impostato un background a gradiente dal bianco al grigio per localizzare al meglio visivamente i contorni dell'immagine da analizzare (come studiato precedentemente). L'area è stata creata per mezzo di quattro livelli sovrapposti, il primo (partendo dal basso) è il contenitore che visualizza l'immagine originale, sopra questo è presente un livello in cui vengono disegnati i *MER* dei batteri selezionati, il terzo

livello viene utilizzato per mostrare un rettangolo di selezione quando si utilizza la selezione rettangolare e il quarto è quello che mostra i risultati dell'elaborazione. I due livelli centrali vengono utilizzati solo nella fase di misurazione, mentre il quarto viene utilizzato in tutte le fasi perché - anche se non visualizza niente sovrapposto all'immagine originale - è quello che riceve gli eventi dal mouse durante la selezione. La finestra di segmentazione ad ogni passo della macchina a stati avvisa l'utente dello stato in corso e gli permette di immettere i parametri per l'elaborazione successiva, durante l'esecuzione delle procedure di *image processing* la *Control Area* e la *Measurement area* vengono oscurate con un'animazione in modo che l'utente non possa generare eventi e si accorga che il programma non è bloccato ma sta lavorando. La finestra si occupa anche di ricevere l'evento di click sui grafici in anteprima per poi aprire la finestra **Filter** e ingrandirli.

L'animazione generata è scritta esclusivamente in *xaml* e consiste in un oscuramento in semi-trasparenza con una scritta "lampeggiante" "*working*". L'animazione viene applicata al parametro *alpha* che controlla l'alpha-blending della scritta in modo da farla variare in modo continuo da 0 a 1 e viceversa, l'evento che fa partire l'animazione è la sua visualizzazione e l'evento che la fa terminare è il nascondimento.

4.2 Interazione delle parti

La *pipeline* di segmentazione inizia con la creazione da parte della **MainApplication** di un oggetto di tipo **ACMeter**, dopo di che viene impostata l'immagine da processare e questo si distacca dal thread principale dando vita alla **SegmentationWindow**, come in figura 4.4. A questo punto l'utente attraverso la **SegmentationWindow** può controllare la macchina a stati di **ACMeter** fino a giungere alla fase di misurazione. In questa fase il ruolo del controllo passa in secondo piano e funge solo da "ponte di comunicazione" tra l'interfaccia e la struttura dell'immagine costruita (i.e. **IImageStructure**), gli eventi generati saranno tutti riconducibili alla scelta di batteri e alla richiesta di loro misure. L'unico evento diverso che può essere generato è la richiesta di ricalcolare le misure con la *funzione di risposta*, in questo caso la struttura dell'immagine dovrà aggiornare i propri valori rical-

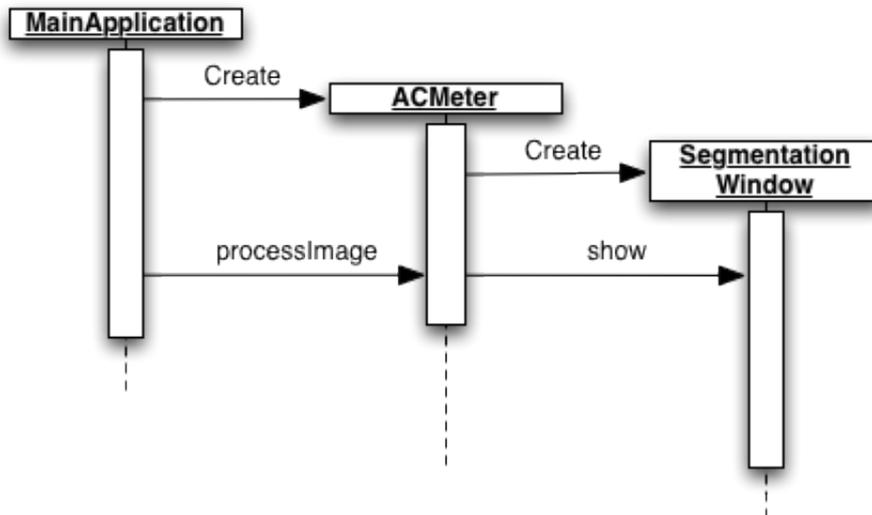


Figura 4.4: Interazione iniziale tra MainApplication e ACMeter. Viene eseguita quando si avvia un nuovo processo di *Image analysis*.

colandoli (questo viene fatto solo una volta e poi le misure vengono tenute in cache, in modo che nelle occasioni successive il sistema sia più pronto nella risposta).

4.3 Dettagli implementativi

In questa sezione verranno analizzati tutti i particolari implementativi che non sono stati discussi nelle fasi precedenti, sia dal punto di vista algoritmico che dal punto di vista della visualizzazione.

4.3.1 Sauvola con integral images

L'algoritmo di *Sauvola* è piuttosto oneroso dal punto di vista computazionale ma può essere migliorato di molto utilizzando le *integral images*. La complessità è dovuta al fatto che è necessario calcolare per una finestra di grandezza arbitraria sia la media che la deviazione standard per ogni vicino. Con le *integral images* questo procedimento viene alleggerito rendendolo indipendente dalla dimensione della

finestra, rimane solo la complessità della creazione dell'immagine integrale che è lineare. Lo stesso accorgimento si può utilizzare per il calcolo della deviazione standard ma, invece che calcolare l'immagine integrale classica, si calcola quella dei quadrati [14].

Integral images

Un'immagine integrale permette di calcolare sommatorie su sottoregioni di un'immagine, è utilizzata in molti algoritmi di *elaborazione delle immagini* per velocizzare i calcoli.

Si supponga di avere un'immagine larga w pixel e alta h pixel, l'immagine integrale sarà larga $w + 1$ pixel e alta $h + 1$ pixel con la prima riga e la prima colonna con solo valori 0, per tutti gli altri pixel i valori saranno dati dalla somma dei pixel nella sotto regione a nord-ovest dell'immagine originale. Di seguito verrà mostrato un esempio con a sinistra l'immagine originale e a destra la rispettiva *integral image*:

$$\begin{bmatrix} 5 & 2 & 3 & 4 & 1 \\ 1 & 5 & 4 & 2 & 3 \\ 2 & 2 & 1 & 3 & 4 \\ 3 & 5 & 6 & 4 & 5 \\ 4 & 1 & 3 & 2 & 6 \end{bmatrix} \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 5 & 7 & 10 & 14 & 15 \\ 0 & 6 & 13 & 20 & 26 & 30 \\ 0 & 8 & 17 & 25 & 34 & 42 \\ 0 & 11 & 25 & 39 & 52 & 65 \\ 0 & 15 & 30 & 47 & 62 & 81 \end{bmatrix}$$

Identificata la regione nell'immagine originale di cui si vuole calcolare la somma, la rispettiva regione nell'immagine integrale è la stessa ma con una colonna in più e una riga in più; la somma è data dai pixel “in basso a destra” + “in alto a sinistra” - “in alto a destra” - “in basso a sinistra”. Per esempio, se si volesse calcolare la somma della regione $[1 : 3, 1 : 4]^2$, la rispettiva regione nell'immagine integrale sarebbe $[1 : 4, 1 : 5]$, a questo punto basterebbe eseguire la semplice operazione $62 + 5 - 14 - 15 = 38$. Come si può vedere il calcolo non dipende più dalla grandezza della regione.

²si considerino gli indici a partire da 0

4.3.2 Aggiornamento interfaccia

L'interfaccia grafica, per come è stata progettata, viene eseguita in un *thread* separato rispetto all'elaborazione dell'immagine, così è necessario che al termine della computazione venga avvisata. In *C#* non è possibile chiamare direttamente dei metodi dell'interfaccia grafica da un *thread* distinto, così è stato necessario definire dei *delegati*³ nella classe `SegmentationWindow`. Questi *delegati* vengono invocati sul *dispatcher* della finestra che poi in base al livello di priorità assegnatogli li esegue.

4.3.3 Particolari dei grafici

La progettazione dei grafici è stata fatta “ex novo” utilizzando i componenti già presenti nell'ambiente *WPF*.

L'area in cui vengono disegnati i grafici (quindi quella compresa tra gli assi) è definita come `Canvas`, qui esistono due metodi per rappresentare i dati: o si utilizza il `Canvas` come se fosse un'area per il disegno oppure si aggiungono elementi come se fosse un gestore di layout. La differenza principale è che nel primo caso i dati vengono visti come un unico disegno, nel secondo caso ogni dato ha una sua entità.

La mia scelta è stata diretta verso la seconda opportunità poiché dovendo implementare la visualizzazione dei valori al passaggio del mouse, questa risultava molto semplificata sfruttando il gestore degli eventi di ogni elemento. L'unica differenza visuale tra i due approcci è che nel primo caso l'*antialiasing* viene applicato alla complessità dei dati mentre nel secondo caso viene applicato ad ogni dato separatamente. Nell'istogramma in anteprima quando le barre sono molto ravvicinate questo porta ad un effetto di gradiente sul colore del grafico. Il problema comunque non disturba la consultazione dei dati.

4.3.4 Implementazione Algoritmi

Alcuni algoritmi utilizzati nella fase di *image processing* sono stati presi da librerie opensource, modificati in base alle esigenze e riutilizzati.

³I delegati sono molto simili a ciò che nel linguaggio *C* viene chiamato puntatore a funzione

La libreria che è stata presa come riferimento è *AForge.NET* [1], sviluppata in *C#*. Le modifiche principali che ho dovuto apportare sono state la cancellazione della direttiva `unsafe`, utilizzata nel linguaggio per poter utilizzare liberamente i puntatori, e la conversione dei tipi di matrici su cui l'algoritmo lavorava in `double`.

Anche il calcolo del *MER* è stato eseguito con una libreria di supporto, questa però non riguarda l'elaborazione delle immagini ma riguarda la gestione e le operazioni di geometria in 2D, per questa è stato necessario convertire la rappresentazione delle coordinate interna al software con quella della libreria.

CAPITOLO 4. IMPLEMENTAZIONE

Capitolo 5

Risultati

Questo capitolo presenterà i risultati ottenuti sia dal punto di vista algoritmico che dal punto di vista dell'interfaccia grafica, quindi la seconda parte sarà una presentazione del prodotto finito. Le immagini dei risultati provengono esclusivamente dall'applicazione e non dai test preliminari effettuati con *MATLAB*.

5.1 Risultati

In figura 5.1 vengono proposti dei particolari di ogni fase di segmentazione per valutarne l'efficienza. Come si può vedere nell'immagine originale spesso i batteri sono molto vicini e difficili da separare, la sola *sogliatura* è inefficace e rivela anche impurità che nelle misure non dovranno essere considerate. Si presti particolare attenzione ai batteri che vengono considerati uniti quando in realtà non lo sono e al batterio a destra che presenta un'impurità a metà del suo corpo, chiaramente non appartenente a sé. Oltre a questo la *sogliatura* produce anche un po' di *sovrasegmentazione* lungo la silhouette. Si può notare come l'*edge detector* rilevi in modo molto migliore dove finisce un batterio e ne inizia un altro, ma spesso produce bordi anche dove non sono necessari. L'unione di questi due risultati dà origine a ciò che viene mostrato nel *labeling*, qui si può notare come le impurità siano state eliminate. In quest'immagine è stato eseguito anche un filtraggio delle aree con superficie minore di 20 pixel, per questo i batteri più piccoli sono stati eliminati dal risultato.

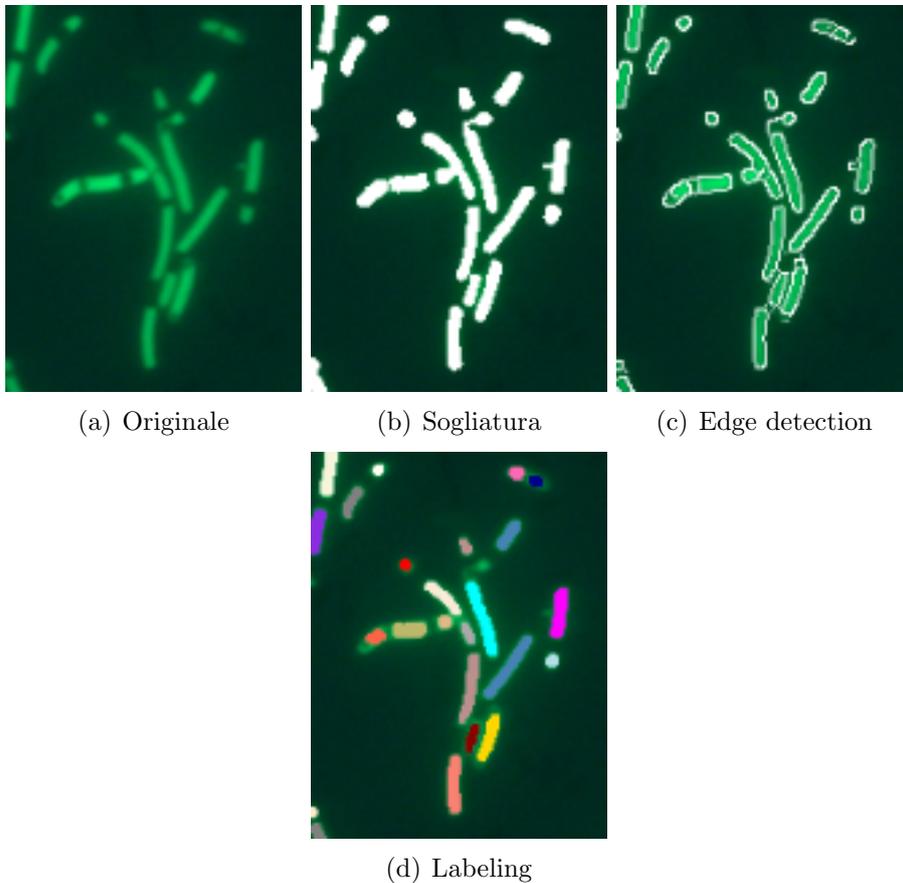


Figura 5.1: Dettagli di acquisizione in tutti i suoi step: originale, sogliatura, edge detection, labeling. Il test è stato eseguito su un'acquisizione con esposizione di 3s.

Le regioni colorate in modo diverso verranno poi considerate come batteri diversi. Si può notare una perfetta coerenza tra il risultato ottenuto e quello atteso. Nella figura 5.2 vediamo un dettaglio del caso in cui i batteri siano in densità molto maggiore. In questo caso la separazione dei batteri è molto più difficile, in certi casi anche l'occhio umano fa fatica a capire dove stia il confine tra uno e l'altro. Come si può vedere dalle immagini la segmentazione produce sempre un risultato piuttosto grossolano, che poi viene lievemente migliorato dall'edge detector. Alla fine della segmentazione il risultato è buono,

per ottenere risultati migliori si sarebbero dovuti impiegare algoritmi molto più complessi, magari basati sul riconoscimento di forme conosciute.

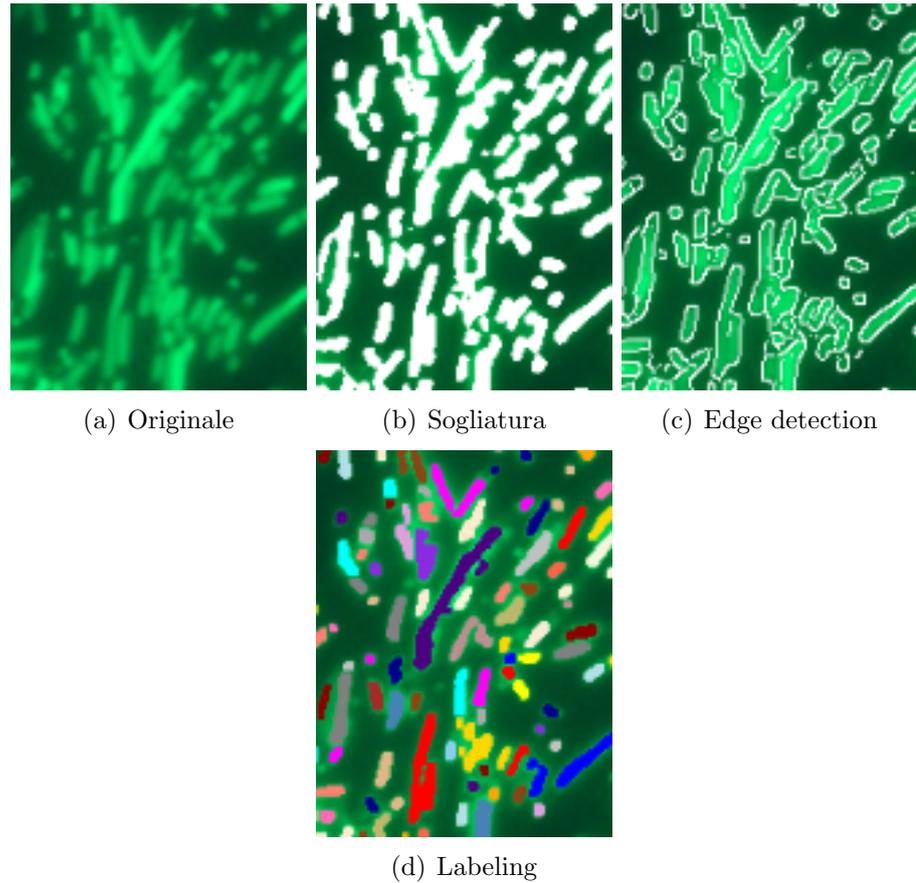


Figura 5.2: Dettagli di acquisizione in tutti i suoi step: originale, sogliatura, edge detection, labeling. Il test è stato eseguito su un'acquisizione con esposizione di 3s.

Con esposizione più bassa il risultato è lievemente migliore ma la trattazione di questo caso rimane un *task* piuttosto duro.

5.2 Presentazione del prodotto finito

L'applicazione si presenta come mostrato in figura 5.3, dove è stato rispettato il layout discusso precedentemente, nella parte destra si susseguono dei pannelli che mostrano le opzioni possibili di volta in volta.

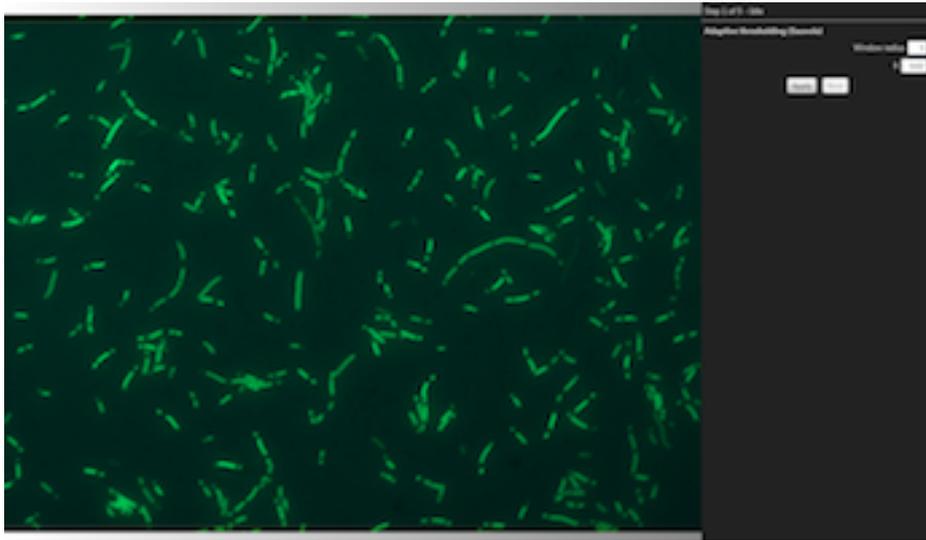


Figura 5.3: Schermata principale dell'applicazione.

Sopra e sotto all'immagine è possibile notare il gradiente messo appunto per valorizzare i bordi di quest'ultima. Il rapporto delle dimensioni dell'immagine rimane costante, quindi scala con la finestra. Ho scelto di non inserire barre di scorrimento in modo che l'utente abbia sempre una vista globale della scena.

Dopo quattro passaggi (il *Refiner* non viene mostrato all'utente poiché privo di parametri) si giunge alla fase di misurazione, che viene mostrata nella figura 5.4. In questa schermata è possibile notare le misure globali ottenute dall'analisi: il primo grafico rappresenta l'istogramma delle sole aree segmentate (i.e. i batteri) mentre il secondo grafico rappresenta la luminosità media di ogni batterio rispetto la sua area. Sotto a questi grafici ne è presente anche un altro che rappresenta la lunghezza di ogni batterio rispetto al rapporto delle dimensioni, è raggiungibile solo attraverso la barra di scorrimento.

Nell'istogramma si può notare quel gradiente che produce un effetto simile ad una texture di cui ho parlato nella sezione 4.3.3. Tra le misure viene anche visualizzato il numero di batteri riconosciuti, in quest'immagine è 618, il cervello umano spesso a colpo d'occhio non riesce a dare una stima del numero di oggetti visti.

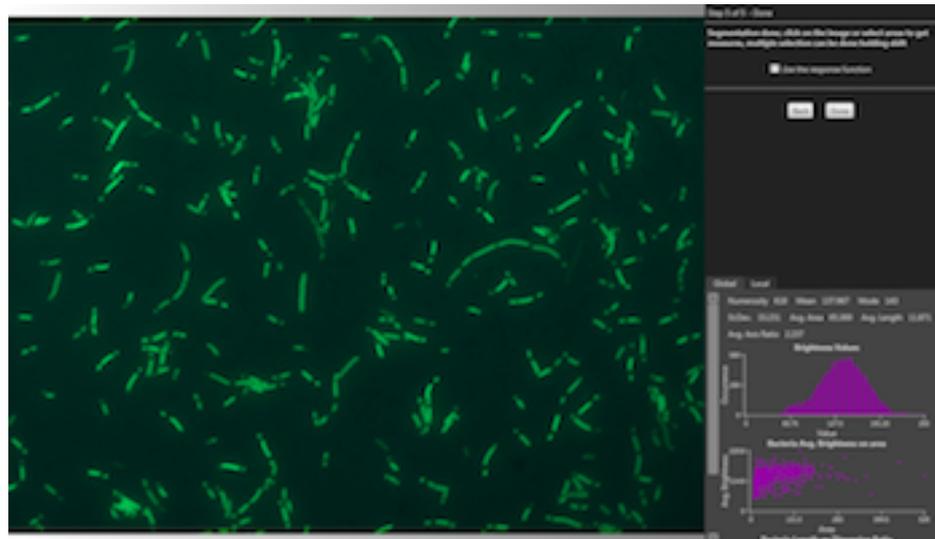


Figura 5.4: Schermata delle misurazioni: misurazioni globali.

Selezionando la scheda delle misurazioni locali sarà possibile abilitare la selezione nell'*Image displaying area*. Qui è disponibile sia la selezione singola che la selezione rettangolare come mostrato in figura 5.5. Premendo *shift* viene abilitata la selezione (o deselegione) multipla.

Come si può notare i grafici non hanno né titolo né dati, questo avviene prima della prima selezione poiché ovviamente ancora non ci sono dati da visualizzare. Il secondo grafico, inoltre, viene visualizzato solo se viene fatta una selezione con più di un batterio.

Nella parte in alto, tra le misure, viene anche visualizzato il numero di batteri selezionati.

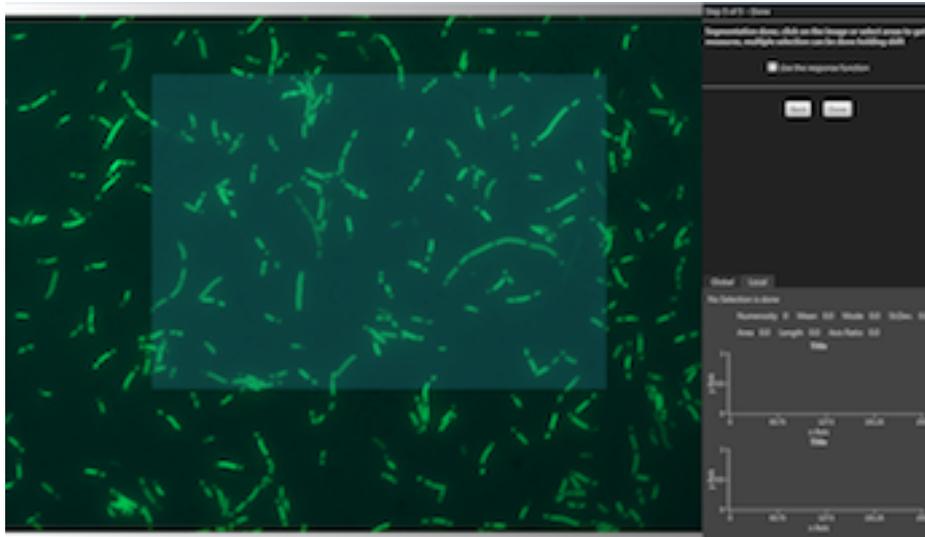


Figura 5.5: Selezione rettangolare sui batteri.

Nella figura 5.6 è possibile notare ciò che accade facendo una selezione rettangolare e una selezione sempre rettangolare ma interna alla prima tenendo premuto shift.

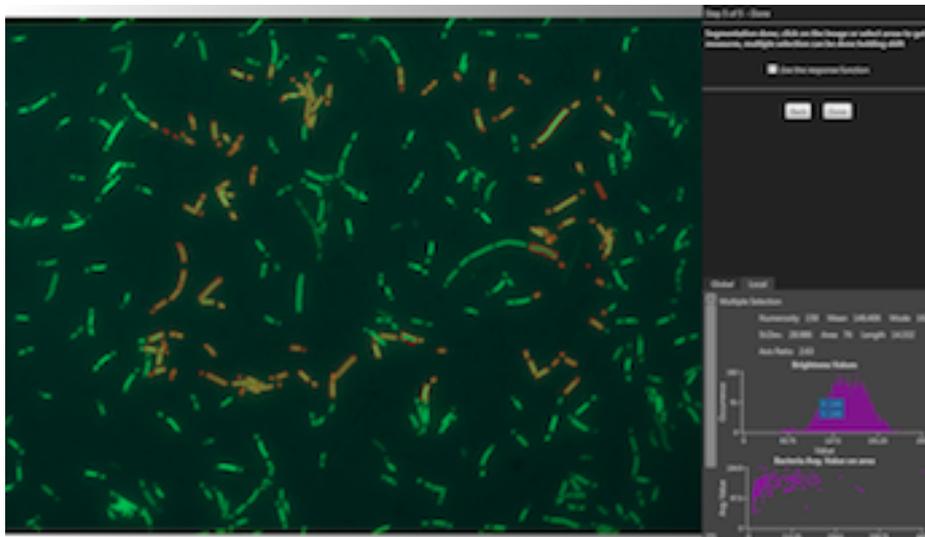


Figura 5.6: Selezione multipla, un rettangolo inscritto in un altro.

La figura 5.6 rivela come sia possibile fare una sottrazione di selezione per rendere l'utente il più libero possibile. Nella parte a destra si può notare che i grafici hanno assunto il valore dei batteri selezionati. Ulteriore particolare di questa figura è l'etichetta che mostra i valori allo scorrere del mouse sopra ai grafici. In figura 5.7 si può vedere un ingrandimento dello stesso istogramma.

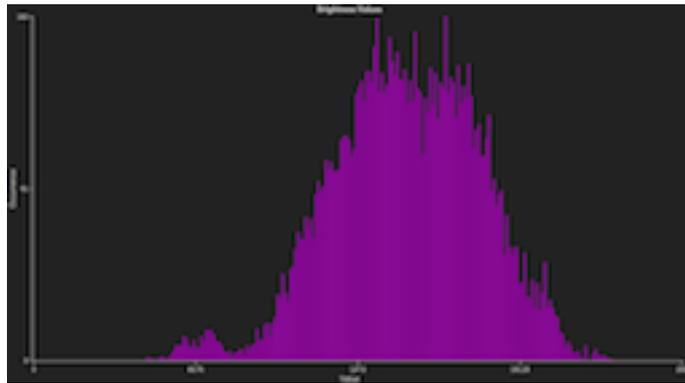


Figura 5.7: Finestra di ingrandimento dell'istogramma.

Nella figura 5.8, invece, si può vedere un ingrandimento di un grafico a dispersione. Il grafico in questo caso è dotato della possibilità di essere filtrato, e quindi di eseguire una selezione inversa sui batteri.

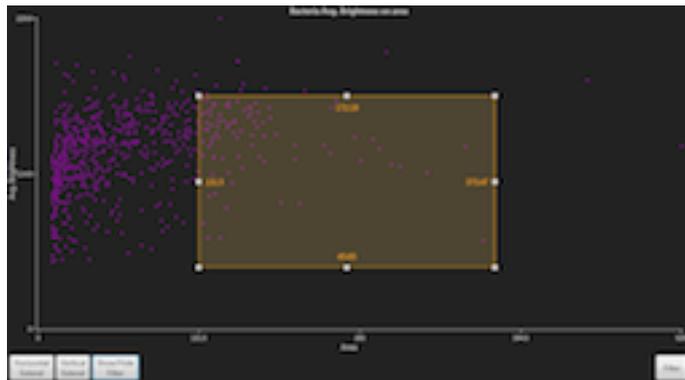


Figura 5.8: Finestra di ingrandimento del grafico a dispersione con rettangolo di filtraggio.

Il rettangolo centrale è quello che permette il filtraggio, può essere visualizzato o nascosto tramite il pulsante *Show/Hide Filter* e mostra su ogni lato il valore a cui il rettangolo eseguirà il filtraggio. Per comodità sono stati aggiunti anche il pulsante *Vertical Extend* e *Horizontal Extend*, questi permettono di utilizzare il rettangolo come se fosse un selettore *a banda*, il vantaggio di non essere realmente a banda - ma un rettangolo - è che può essere utilizzato in entrambe le direzioni. Per applicare il filtro selezionato è sufficiente premere il tasto in basso a destra *Filter*.

Capitolo 6

Conclusioni e possibili sviluppi

Il lavoro presentato in questa tesi, in un contesto di ricerca interdisciplinare nel campo della biologia e dell'ingegneria biomedica e informatica, è stato mirato alla realizzazione di un software dotato di interfaccia grafica per consentire l'analisi quantitativa di colture di batteri geneticamente modificati. La misura richiesta su questo tipo di colture avviene mediante un microscopio a fluorescenza provvisto di telecamera di acquisizione. Per consentire una misura affidabile e comparabile tra le immagini acquisite a tempi di esposizione diversi, l'analisi è slegata dalla intensità dei valori dei pixel, bensì riportata in un dominio radiometrico (a meno di una costante ed un fattore di proporzionalità) attraverso la conoscenza della funzione di risposta della telecamera.

La fase di misura è stata progettata per consentire all'utente un sistema guidato (*wizard*), che conduce l'utente nelle varie fasi di *image processing*, necessarie alla precisa individuazione dei batteri presenti nelle immagini; dalla scelta dei parametri degli algoritmi di segmentazione alla visualizzazione del risultato finale. Particolare attenzione è stata posta nella progettazione del software dal punto di vista della fruibilità e della presentazione dei dati agli utenti, in questo caso, biologi, spesso privi di una formazione informatica nel campo dell'elaborazione di immagini.

Strumenti disponibili in commercio per questo tipo di misure, quali i fluorimetri, non consentono all'utente un'analisi a livello di singoli batteri o gruppi di essi, ma limitano l'analisi all'intero campione. Il software realizzato in questo lavoro, trova quindi impiego in tutti quei campi della biologia sintetica dove è richiesta la verifica e l'analisi della modifica del DNA, quantificandone gli effetti anche a carattere locale, come ad esempio la quantificazione del numero di batteri che hanno espresso le caratteristiche richieste. L'utilizzo di tecniche di elaborazione ed analisi di immagini in questo dominio applicativo consente di ottenere benefici sia da un punto di vista della misurazione in sé, permettendo la quantificazione delle caratteristiche dei batteri non solo da un punto di vista statistico, sia benefici in termini economici, dato il costo degli strumenti concorrenti.

Possibili sviluppi correlati a questo lavoro potrebbero riguardare l'utilizzo di algoritmi di *object recognition* per migliorare la segmentazione dei batteri, soprattutto per immagini ad alta densità, al fine di migliorare la precisione delle misure sui singoli batteri.

Appendice A

Sviluppo e calcolo della funzione di risposta

A.1 Sviluppo

Partendo dalla funzione di errore:

$$\varepsilon = \sum_{q=1}^{Q-1} \sum_{p=1}^P \left(\sum_{n=0}^N c_n M_{p,q}^n - R_{q,q+1} \sum_{n=0}^N c_n M_{p,q+1}^n \right)^2$$

per comodità si raggruppa la sommatoria

$$\varepsilon = \sum_{q=1}^{Q-1} \sum_{p=1}^P \left(\sum_{n=0}^N c_n M_{p,q}^n - R_{q,q+1} c_n M_{p,q+1}^n \right)^2$$

a questo punto è necessario calcolare la derivata

$$\frac{\partial \varepsilon}{\partial c_n} = 2 \left(\sum_{n=0}^N c_n M_{p,q}^n - R_{q,q+1} c_n M_{p,q+1}^n \right) (M_{p,q}^n - R_{q,q+1} M_{p,q+1}^n)$$

d'ora in poi per brevità utilizzerò

$$\alpha_{p,q}^n = (M_{p,q}^n - R_{q,q+1} M_{p,q+1}^n)$$

Data la derivata vengono semplificate le costanti e viene messa in forma matriciale per essere risolta come sistema lineare $Ax = B$ dove

A ha un numero di righe pari al numero di pixel di un'immagine P (eventualmente moltiplicato per Q se si fa con più di 2 immagini) e un numero di colonne pari al grado N del polinomio. La matrice risultante per un'immagine di dimensioni 1280×960 dopo aver inserito anche il vincolo $c_N = I_{max} - \sum_{n=0}^{N-1} c_n$ è

$$A = \begin{pmatrix} \alpha_{p,q}^0 - \alpha_{p,q}^N & \cdots & \alpha_{p,q}^{N-1} - \alpha_{p,q}^N \\ \alpha_{p+1,q}^0 - \alpha_{p+1,q}^N & \cdots & \alpha_{p+1,q}^{N-1} - \alpha_{p+1,q}^N \\ \vdots & \vdots & \vdots \end{pmatrix}$$

$$x = \begin{pmatrix} c_0 \\ c_1 \\ \vdots \\ c_{N-1} \end{pmatrix}$$

$$B = \begin{pmatrix} -\alpha_{p,q}^N \\ \vdots \\ -\alpha_{960*1280,q}^N \end{pmatrix}$$

A questo punto il sistema sarà facilmente risolvibile come $x = B/A$.

A.2 Implementazione

Il codice *MATLAB* per trovare i coefficienti è stato diviso in 3 parti:

alphaVect.m è una funzione che computa una matrice di dimensione $1 \times P \cdot Q$ in cui in ogni riga c'è un valore $\alpha_{p,q}^n$ dove n viene passato come parametro.

cleaningSaturation.m è una funzione che ripulisce le immagini dai pixel più vicini alla saturazione del 3% del range dei valori e controlla anche il vincolo di monotonicità.

CoefficientsEval.m è la funzione principale che calcola i coefficienti della funzione di risposta e li restituisce in ordine inverso.

APPENDICE A. SVILUPPO E CALCOLO DELLA FUNZIONE DI RISPOSTA

Alla funzione `CoefficientsEval.m` è necessario passare un'array di 3 dimensioni in cui le prime due dimensioni rappresentano l'immagine e la terza dimensione viene utilizzata per separare le varie immagini, in altre parole se si volesse vedere in 3D sarebbe un array di immagini complanari.

Di seguito il codice delle tre funzioni:

```
function value = alphaVect(Images, Exps, n)
    Col = [];
    for kk = 1:size(Images,2)-1
        R = Exps(kk)/Exps(kk+1);
        Im1Vect = Images(:,kk);
        Im2Vect = Images(:,kk+1);
        Col = cat(1,
            Col,
            (Im1Vect./double(255)).^n -
            R.*((Im2Vect./double(255)).^n));
    end
    value = Col;
end

function value = cleaningSaturation(Images)
    %creating one column for each image
    VectorImage = zeros(size(Images,1)*size(Images,2),
        size(Images,3));
    for ii = 1:size(Images,3)
        Image = Images(:,:,ii);
        VectorImage(:,ii) = Image(:);
    end

    SaturatedPixel = true(size(VectorImage,1),1);
    for ii = 1:size(VectorImage, 2)
        SaturatedPixel =
            SaturatedPixel & (VectorImage(:,ii) < 248);
        SaturatedPixel =
            SaturatedPixel & (VectorImage(:,ii) > 7);
    end
end
```

*APPENDICE A. SVILUPPO E CALCOLO DELLA FUNZIONE
DI RISPOSTA*

```
end

Cleaned = VectorImage(SaturatedPixel,:);
%Monotonic Constraint
Monotonic = true(size(Cleaned,1),1);
for ii = 1:size(Cleaned,2)-1
    Monotonic =
        Monotonic & (Cleaned(:,ii) < Cleaned(:,ii+1));
end

value = Cleaned(Monotonic,:);

end

function Coeff = CoefficientsEval(Images, Exps, N)
    Images2 = zeros(size(Images));
    %Apply gaussian filter to each image
    for ii = 1:size(Images,3);
        Kernel = imdivide(double(
[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]),
double(273));
        Images2(:, :, ii) =
            imfilter(Images(:, :, ii), Kernel, 'conv');
    end

    %Cleaning saturation data
    size(Images2)
    Cleaned = cleaningSaturation(Images2);
    size(Cleaned)
```

*APPENDICE A. SVILUPPO E CALCOLO DELLA FUNZIONE
DI RISPOSTA*

```
%Creating A Matrix
A=[];

%Creating A matrix
for ii = 0:N-1
    A = cat(2, A, alphaVect(Cleaned, Exps, ii));
end
VN = -alphaVect(Cleaned, Exps, N);
AN = [];
for ii = 0:N-1
    AN = cat(2, AN, VN);
end
A = A + AN;

%Vector B
B = VN;

%Calculating Coeffs from 0 to N-1
Coeff1 = A \ B;
%Calculating Coeff N
Coeff = cat(1,Coeff1, 1-sum(Coeff1));

end
```

*APPENDICE A. SVILUPPO E CALCOLO DELLA FUNZIONE
DI RISPOSTA*

Ringraziamenti

Innanzitutto i miei ringraziamenti vanno sicuramente al Prof. Alessandro Bevilacqua e all'Ing. Alessandro Gherardi che mi hanno seguito assiduamente durante tutto lo sviluppo della tesi e sono stati sempre molto disponibili nonostante le innumerevoli e-mail inviategli. Un ulteriore ringraziamento va al Prof. Emanuele Domenico Giordano e al Dott. Filippo Piccinini che mi hanno supportato in tutto ciò che riguarda la parte biomedica/biologica.

Al secondo posto, ma non per ordine di importanza, ringrazio i miei genitori che mi hanno permesso di affrontare e concludere gli studi con innumerevoli sforzi e spesso sacrificando la loro vita per fornirne una migliore a me.

Per i due colleghi di studio più affiatati non necessitano neanche ringraziamenti perché loro già sanno quanto sia stato importante il supporto che ci siamo dati in questi 5 anni, in ogni caso vi ringrazio Lazza e Mola (Non vi ho mai chiamati per nome proprio e non lo farò di certo ora). Spero con voi di poter condividere anche situazioni lavorative e comunque di mantenere un rapporto di amicizia il più a lungo possibile.

Per concludere, ringrazio anche gli amici più stretti: Mattia, Aurelio e Gippo, la mia ragazza Uma e mia sorella Erika che oltre ad aver sempre creduto in me e ad avermi supportato, hanno sempre offerto una spalla su cui poter contare.

Bibliografia

- [1] AForge.NET. open source framework for ai and cv, available online: <http://www.aforge.net.com/framework/>.
- [2] A. Bevilacqua, A. Gherardi, and L. Carozza. A robust approach to reconstruct experimentally the camera response function. *Image Processing Theory, Tools and Applications, 2008. IPTA 2008. First Workshops on*, 2008.
- [3] F. Dai, M. Sugisaka, and B. Zhang. *Image segmentation*, chapter A Survey of Image Segmentation by the Classical Method and Resonance Algorithm. InTech, 2011.
- [4] R. Fisher, S. Perkins, A. Walker, and E. Wolfart.
- [5] D. B. Goldman and J.-H. Chen. Vignette and exposure calibration and compensation. In *The 10th IEEE International Conference on Computer Vision*, pages 899–906, Oct. 2005.
- [6] M. J. Langford. *Basic photography*. Focal Press, 2000.
- [7] T. Mitsunaga and S. K. Nayar. Radiometric self calibration. *Computer Vision and Pattern Recognition, 1999. IEEE Computer Society Conference on.*, 1999.
- [8] J. G. Morin and J. W. Hastings. Energy transfer in a bioluminescent system. *Journal of Cellular Physiology*, 77(3):313–318, 1971.
- [9] C. A. Poynton. *Digital video and HDTV: algorithms and interfaces*. Morgan Kaufmann, 2003.

- [10] D. C. Prasher, V. K. Eckenrode, W. W. Ward, F. G. Prendergast, and M. J. Cormier. Primary structure of the aequorea victoria green-fluorescent protein. *Gene*, 111(2):229 – 233, 1992.
- [11] R. W. Pridmore. Complementary colors theory of color vision: Physiology, color mixture, color constancy and color perception. *Color Research and Application*, 36(6):394–412, 2011.
- [12] J. C. Russ. *The Image Processing Handbook Sixth Edition*, pages 513, 514, 515. North Carolina State University, 2011.
- [13] G. Selvaggio. Elaborazione di immagini di microscopia ottica per la stima del livello di espressione di una proteina fluorescente in singola cellula. Master’s thesis, Alma Mater Studiorum, Università di Bologna, seconda facoltà di ingegneria con sede a Cesena, 2009.
- [14] F. Shafaita, D. Keysersa, and T. M. Breuelb. Efficient implementation of local adaptive thresholding techniques using integral images. *Document Recognition and Retrieval XV*, 2008.
- [15] O. Shimomura, F. H. Johnson, and Y. Saiga. Extraction, purification and properties of aequorin, a bioluminescent protein from the luminous hydromedusan, aequorea. *Journal of Cellular and Comparative Physiology*, 59(3):223–239, 1962.
- [16] D. Skoog, F. Holler, and T. Nieman. *Principles of Instrumental Analysis*. Saunders College Publishing, New York, 1997.
- [17] K. R. Spring and M. W. Davidson. Introduction to fluorescence microscopy. *Nikon MicroscopyU*, 2008.
- [18] D. R. Szeliski. *Computer Vision, Algorithms and Applications*, page 413. Springer London Dordrecht Heidelberg New York, 2010.

Elenco delle figure

1.1	Stilizzazione di un microscopio ad epifluorescenza . . .	6
1.2	Sinistra: immagine di un muro a $f/1.4$. Centro: lo stesso muro a $f/5.6$, dimostra che il vignetting ottico diminuisce con la dimensione dell'apertura. Destra: la forma della pupilla di entrata varia sia con l'apertura di entrata che con l'angolo di incidenza. [5]	12
1.3	Pipeline per l'acquisizione di immagini, ognuno degli stage visualizzati introduce non-linearità o rumore [18]	13
1.4	Suddivisione del fascio di luce nei tre colori fondamentali additivi per mezzo di filtri tricoici.	14
3.1	Dettaglio di microscopia in fluorescenza di batteri geneticamente modificati sottoposti al GFP, intensità del canale verde alle esposizioni specificate con rispettivo istogramma delle intensità.	35
3.2	Rappresentazione in 3D dell'immagine intera con esposizione 12s	37
3.3	Particolare di segmentazione con algoritmo di risonanza	38
3.4	Finestra di base utilizzata per la segmentazione	44
3.5	Rapporto tra i raggi delle finestre	45
3.6	Distribuzione del parametro k	47
3.7	Macchina a stati del motore di elaborazione	49
3.8	Layout della finestra principale	51
3.9	Macchina a stati per la selezione dei batteri	54
3.10	Colori complementari: sono detti complementari i colori opposti nel cerchio (ovvero con tinta opposta). . .	56

4.1	Struttura interna del namespace <code>ImagingChart</code> . Classi pubbliche con la linea continua e classi private con la linea tratteggiata.	59
4.2	Struttura interna del namespace <code>Measurement</code> , sono rappresentati solo i namespace di secondo livello, quelli di terzo livello sono mostrati in figura 4.3. Classi pubbliche con la linea continua e classi private con la linea tratteggiata.	62
4.3	Struttura interna del namespace <code>Measurement</code> , namespace di terzo livello. Classi pubbliche con la linea continua e classi private con la linea tratteggiata. . . .	63
4.4	Interazione iniziale tra <code>MainApplication</code> e <code>ACMeter</code> . Viene eseguita quando si avvia un nuovo processo di <i>Image analysis</i>	66
5.1	Dettagli di acquisizione in tutti i suoi step: originale, sogliatura, edge detection, labeling. Il test è stato eseguito su un'acquisizione con esposizione di 3s.	72
5.2	Dettagli di acquisizione in tutti i suoi step: originale, sogliatura, edge detection, labeling. Il test è stato eseguito su un'acquisizione con esposizione di 3s.	73
5.3	Schermata principale dell'applicazione.	74
5.4	Schermata delle misurazioni: misurazioni globali.	75
5.5	Selezione rettangolare sui batteri.	76
5.6	Selezione multipla, un rettangolo inscritto in un altro.	76
5.7	Finestra di ingrandimento dell'istogramma.	77
5.8	Finestra di ingrandimento del grafico a dispersione con rettangolo di filtraggio.	77