Alma Mater Studiorum — Università di Bologna
Campus di Cesena

Dipartimento di Informatica — Scienza e Ingegneria
Corso di Laurea triennale in Ingegneria e Scienze Informatiche

DS4H Image Alignment: Modulo per allineamento multimodale automatico considerando deformazioni solide

Tesi in Programmazione

Presentata da:
Marco Edoardo Duma
Matricola 694493

Relatore:
Prof. Antonella Carbonaro

Correlatori:
Prof. Filippo Piccinini
Prof. Giovanni Martinelli
Prof. Gastone Castellani

III sessione di laurea
Anno Accademico 2021–2022

Parole chiave

Istologia

Microscopia

Immunoistochimica sequenziale

Allineamento multi-modale

Fiji plugin

Sommario

(EN) Most of the time, the deep analysis of a biological sample requires the acquisition of images at different time points, using different modalities and/or different stainings. These information give functional and morphological insights, but to really exploit the images acquired they must be co-registered to be then able to proceed with colocalization analysis. Practically speaking, accordingly to the Aristotle's principle "The whole is greater than the sum of its parts", multi-modal image registration is the challenging task that brings to fuse together complementary signals. In the last years, several methods for image registration have been described in the literature, but unfortunately there is not one method that works for all applications. In addition, today there is no user-friendly tool for aligning images without any computer skills. In this work, besides revising all the solutions freely available for co-registering microscopy images, we describe *DS4H Image Alignment*, an open-source *Fiji* plugin for aligning multimodality, immunohistochemistry, and/or immunofluorescence 2D microscopy images, designed with the goal to be extremely easy-to-use.

(IT) Il più delle volte, l'analisi approfondita di un campione biologico richiede l'acquisizione di immagini in differenti momenti, utilizzando differenti modalità e/o colorazioni. Le informazioni acquisite forniscono dati funzionali e morfologici, ma per sfruttare realmente tutta la conoscenza, le immagini acquisite devono essere co-registrate per poter poi procedere con analisi di co-localizzazione. In pratica, secondo il principio aristotelico "Il tutto è maggiore della somma delle sue parti", la registrazione multimodale dell'immagine è un compito impegnativo che porta a fondere insieme segnali complementari. Negli ultimi anni, sono stati descritti diversi metodi per la registrazione delle immagini, ma sfortunatamente non esiste un singolo metodo che funziona per tutte le applicazioni. Inoltre, oggi non esiste uno strumento intuitivo per l'allineamento delle immagini che non richieda importanti competenze informatiche da parte dell'utente. In questo lavoro, oltre a revisionare tutte le soluzioni disponibili gratuitamente per la co-registrazione di immagini di microscopia, descriviamo *DS4H Image Alignment*, un plug-in open source sviluppato per *Fiji* per l'allineamento di immagini 2D multimodali, di immunoistochimica e/o di immunofluorescenza, progettato con l'obiettivo di essere estremamente facile da utilizzare.

Indice

Introduzione

Il progetto di Tesi *"DS4H Image Alignment"* è stato sviluppato all'interno del gruppo di ricerca *"Data Science for Health"* (DS4H), gruppo che raccoglie professori e ricercatori del *"IRCCS Istituto Romagnolo per lo Studio dei Tumori Dino Amadori"* (IRST) di Meldola (FC) e dell'*Università di Bologna* (UniBo), professionisti attivi nel proporre metodi ed applicativi informatici per risolvere problemi aperti nel campo della microscopia e collegati al mondo della medicina, biologia e fisica. In particolare, per questo progetto di Tesi, la Prof.ssa Antonella Carbonaro e il Dr. Filippo Piccinini hanno sollevato il problema e proposto le prime soluzioni teoriche, e il Professore Giovanni Martinelli e il Professore Gastone Castellani hanno consentito l'acquisizione delle immagini utilizzate partendo dalla analisi di provini istologici di pazienti.
DS4H Image Alignment è un software nato con l'intento di fornire un applicativo estremamente user-friendly che permette ai ricercatori di allineare immagini 2D di provini istologici provenienti da pazienti, al fine di valutare con semplicità cross-correlazione tra differenti marcatori tumorali. Occorre specificare che ad IRST ogni giorno i medici, biologi, fisici analizzano provini istologici applicando differenti marcatori fluorescenti per individuare differenti caratteristiche tumorali. Da questi provini vengono tipicamente acquisite immagini, dove il soggetto di base è lo stesso (*i.e.* il tessuto, la biopsia), ma l'oggetto marcato è differente (ad esempio, in alcune immagini possono essere usati dei marcatori per il nucleo delle cellule, in altre dei marcatori per la membrana, in altre dei marcatori per il citoplasma). È quindi utile poter disporre di uno strumento che riesca a fornire un'immagine unica che mostri allineate tutte le precedenti per poter apprezzare le caratteristiche del tessuto sovrapponendo le informazioni riportate nelle singole immagini precedentemente acquisite.
Il progetto *DS4H Image Alignment* è iniziato nel 2018 con la Tesi in Web Semantico di Stefano Belli [1], poi diffusa attraverso l'articolo scientifico: *J. Bulgarelli, M. Tazzari, A.M. Granato, L Ridolfi, S. Maiocchi, F. De Rosa, M. Petrini, E. Pancisi, G. Gentili, B. Vergani, F. PICCININI, A. CARBONARO, B.E. Leone, G. Foschi, V. Ancarani, M. Framarini, M. Guidoboni, Dendritic cell vaccination in metastatic melanoma turns "non-T cell inflamed" into "T-cell inflamed" tumors, Frontiers in Immunology, 2019. IF2019: 4.716. DOI: 10.3389/fimmu.2019.02353.* [2]
La prima versione di *DS4H Image Alignment* prevedeva metodiche per l'allineamento manuale di immagini 2D di provini istologici. Le immagini facevano riferimento allo stesso provino ma erano relative a differenti marcatori fluorescenti. Per questo motivo il tool è stato

descritto come un applicativo per registrazione multi-modale manuale. In pratica, *DS4H Image Alignment 1.0* è in grado di gestire immagini di dimensione e scala differente, ma non presenta nessuna metodica per l'allineamento automatico. L'utente doveva definire punti di interesse corrispondenti in ciascuna immagine. Invece, la versione *1.0* del software ha come obiettivo principale quello di limitare al massimo il numero di interventi a carico dell'utente e rendere il processo di allineamento delle immagini quanto più automatico possibile.
Nei seguenti capitoli verrà descritto il contesto del progetto e verrà illustrata nei dettagli la soluzione proposta per l'allineamento multi-modale automatico. In particolare, nel Capitolo 1 si dettagliano alcuni cenni di immunoistochimica per meglio comprendere la generazione di immagini multimodali utilizzando differenti marcatori fluorescenti. Nel Capitolo 2 si introduce l'ambiente di sviluppo e distribuzione del software, meglio definito come plugin/modulo. In particolare, è stato scelto di distribuire *DS4H Image Alignment* come plugin di *Fiji*, una versione specifica di *ImageJ* molto diffusa tra i ricercatori. Nel Capitolo 3 viene descritto in dettaglio *DS4H Image Alignment*, in modo da riuscire a comprendere meglio quali sono le mancanze poi superate nella successiva versione. *DS4H Image Alignment 1.0* è stato poi confrontato con i tool già presenti in letteratura, descritti nel Capitolo 4. Infine, nel Capitolo 5 viene descritto il modulo per l'allineamento multi-modale automatico cuore di questo progetto di Tesi, poi riassunto nel rimanente Capitolo.

1 Allineamento multi-modale per istologia

L'istologia è il ramo delle discipline biologiche che studia la struttura microscopica e ultramicroscopica dei tessuti e degli organi animali e vegetali dal punto di vista morfologico, istochimico e delle attività funzionali da essi applicate [3].Tipicamente, si parte dalla analisi di un campione biologico o tessuto prelevato da paziente e tramite una sequenza di operazioni di preparazione si conducono analisi microscopiche per studiarne le caratteristiche normo-patologiche ed identificare malattie o problematiche in generale.

1.1 Preparazione istologica

La preparazione istologica è basata su 5 step fondamentali:

1. Fissazione
2. Disidratazione
3. Inclusione
4. Sezionamento
5. Colorazione

Il primo passo per produrre un campione istologico è quello della fissazione. Questa procedura è essenziale poiché comprende tecniche di conservazione del tessuto, tecniche atte a rafforzare la struttura cellulare ed infine una valutazione approfondita del campione per determinare se dev'essere esposto ad antigeni, in dipendenza dallo scopo per il quale è stato acquisito dello studio per il quale è stato acquisito [4].
I fissativi più comuni sono a base di formaldeide, come ad esempio la formalina neutra tamponata (NBF) adoperata per preservare i tessuti o la paraffin-formalina (PFA) adoperata nel contesto dell'immunoistochimica. A seguito della fissazione si procede con la disidratazione fatta tipicamente tramite etanolo, e la rimozione dell'alcol tramite xylene. Questa serie di operazioni serve a preparare i tessuti e rendere semplice la fase di sezionamento.

L'ultimo passaggio prima del sezionamento è quella dell'inclusione, che viene effettuata di solito tramite paraffina. Questo processo può portare ad un deterioramento delle cellule, a causa della prolungata esposizione al calore, e per evitare che ciò accada si congelano i tessuti dopo questa fase.
La fase di sezionamento concerne la preparazione di sezioni mediante microtomo. Le sezioni ottenute sono generalmente, molto sottili perché la luce dei microscopi possa passarci attraverso e permettere l'analisi da parte del ricercatore o specialista. L'ultima fase è quella della colorazione ed andremo ad analizzarla in dettaglio nella prossima sezione.

1.2 Co-registrazione multi-modale

Durante la preparazione di un campione istologico vengono tipicamente usate più colorazioni per rivelare proprietà distinte del tessuto o di differenti componenti dello stesso. A causa dei lavaggi, e dei vari processi utili a completare ogni singola colorazione, il tessuto stesso viene però tipicamente deformato ed è richiesta quindi una registrazione non rigida delle varie immagini poi acquisite per poter procedere con una analisi globale del campione [5]. Definiamo quindi con il termine "registrazione multi-modale" l'acquisizione e successivo allineamento di immagini provenienti dallo stesso campione ma che presentano delle differenze in termini di marcatura/colorazioni. Un esempio di queste marcature differenti è mostrato nelle Figure da 1.1 a 1.4 che rappresentano immagini, per la maggior parte acquisite in fluorescenza. L'immagine Figura 1.1 è stata colorata con Cy3, o Cianina, colorante visibile anche occhio nudo, che mette in evidenza il citoplasma della cellula. Per la Figura 1.2, è stato usato il DAPI, o "4', 6-diamidin-2-fenilindolo", è utile ad evidenziare i nuclei, per la Figura 1.3 il FITC, o Isotiocianato di fluoresceina, per le membrane ed infine Figura 1.4 usando la luce classica.

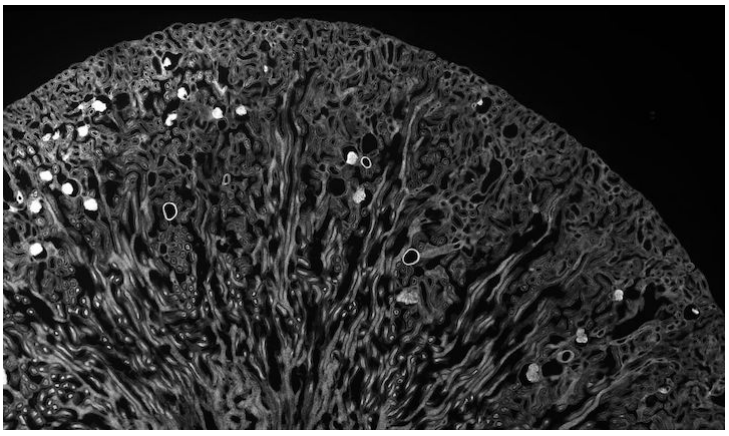

Figura 1.1: Esempio di campione colorato con Cy3

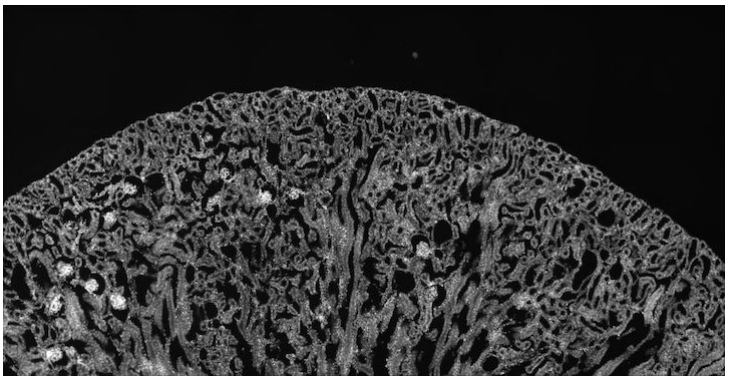

Figura 1.2: Esempio di campione colorato con DAPI

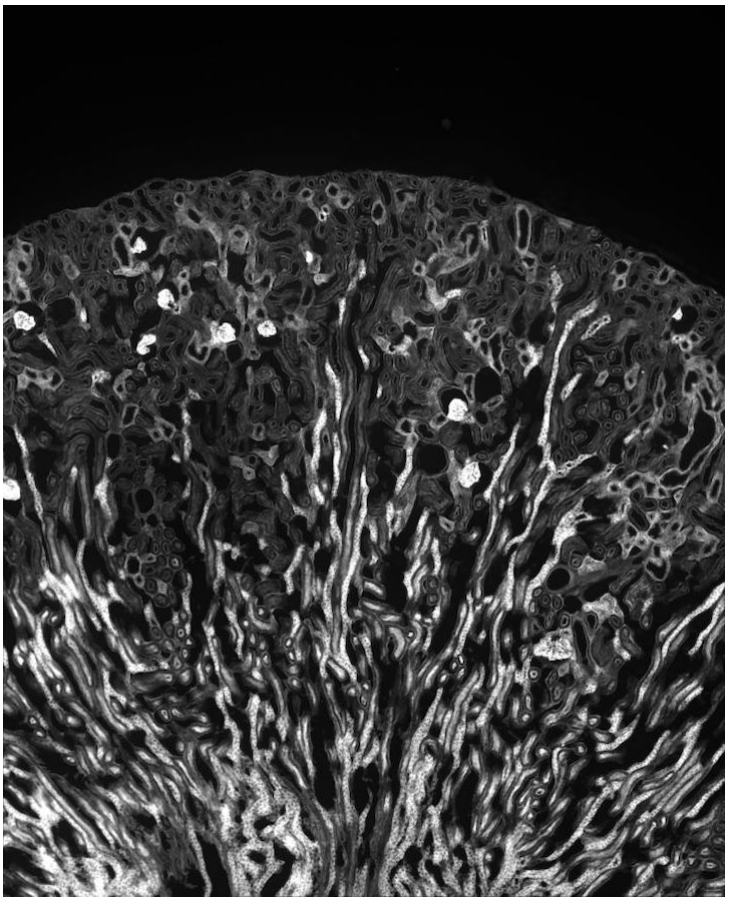

Figura 1.3: Esempio di campione colorato con FITC

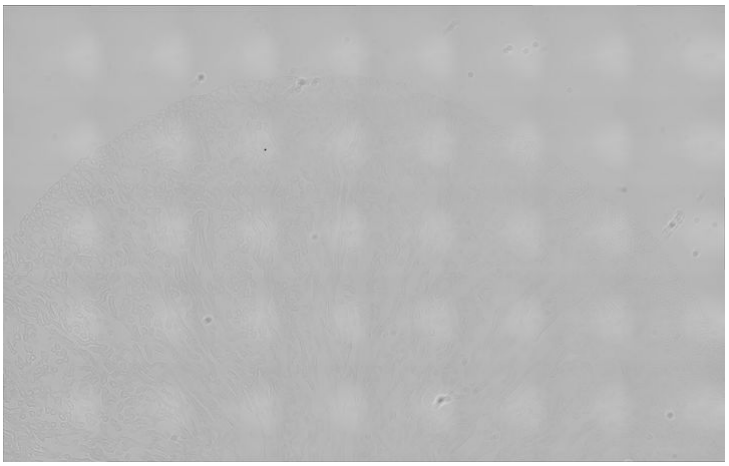

Figura 1.4: Esempio di campione sottoposto a luce classica

1.3 Algoritmi di co-registrazione automatica

Per poter portare a compimento la co-registrazione multi-modale occorre utilizzare algoritmi di *Computer Vision* in grado di occuparsi di questi cinque punti:

1. Rilevamento delle caratteristiche salienti dell'immagine;
2. Definizione della corrispondenza delle caratteristiche tra diverse immagini;
3. Rilevamento delle anomalie;
4. Derivazione della funzione di trasformazione;
5. Ricostruzione dell'immagine finale.

Chiariamo preventivamente che quando si utilizza il termine *descrittore* si fa riferimento al contenitore che altro non è che il contenente di tutte le informazioni ottenute dopo l'elaborazione di un immagine. Di seguito descriveremo brevemente alcuni esempi di algoritmi noti per la registrazione di immagini, partendo da algoritmi, datati e dispendiosi in termini di risorse, fino a raggiungere quelli più recenti e performanti.

1.3.1 SIFT - Scale Invariant Feature Transform

L'algoritmo *"SIFT"* (*Scale Invariant Feature Transform*) è stato realizzato da D.G. Lowe nel 1999 [6]. Il rilevamento di corrispondenze tra immagini, è basato sull'algoritmo chiamato *"Differenze di Gaussiane"*, abbreviato con l'acronimo DoG. Le DoG sono usate per trovare i punti d'interesse invarianti rispetto alla scala e all'orientamento. Per ogni agglomerato di punti si ricerca un modello adatto a determinarne posizione e scala, definendo punti chiave basati sulla stabilità delle loro misurazioni. Per ogni punto chiave viene assegnato un orientamento, in base alla direzione del gradiente locale dell'immagine. I gradienti locali

delle immagini vengono misurati ad una scala selezionata, e vengono trasformati in una rappresentazione che permette di considerare cambi d'illuminazione e distorsione della forma.
La rappresentazione spazio-scala di un immagine è definita da una funzione L(x, y, σ) che è prodotta dalla convoluzione di una variabile spazio-scala Gaussiana G con un immagine in input I [6].

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y) \tag{1.1}$$

dove la funzione G è definita da:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x2+y2)}{2\sigma^2}} \tag{1.2}$$

Al fine di individuare le collocazioni dei punti chiave stabili nella rappresentazione spazio-scala, esprimiamo una funzione D(x, y, σ) e calcoliamo la correlazione tra due scale vicine, facendo la differenza applicando al primo una costante moltiplicativa k:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \tag{1.3}$$

$$= L(x, y, k\sigma) - L(x, y, \sigma) \tag{1.4}$$

1.3.2 SURF - Speeded Up Robust Feature

L'algoritmo *"SURF"* (*Speeded Up Robust Feature*) è nato nel 2006 [7], come *SIFT*, è basato sull'analisi della rappresentazione spazio-scala Gaussiana, ma si basa sul determinante dato da una matrice Hessiana. Grazie all'utilizzo di immagini integrali riesce a velocizzare il processo di identificazione di punti d'interesse. Questo lo rende, meno dispendioso a livello computazionale rispetto a SIFT.
Il concetto di immagine integrale espressa in forma matematica è data da I_Σ ad una posizione $\mathbf{x} = (x, y)^T$, ed è la somma di tutti i pixel presenti all'interno di una porzione rettangolare di un'immagine presa in input:

$$I_\Sigma = \sum_{i=0}^{i \leq x} \sum_{j=0}^{i \leq y} I(i, j) \tag{1.5}$$

Un Hessiana è una matrice quadrata le cui componenti sono derivate parziali seconde di una funzione, che nel caso di questo algoritmo ha questa forma:

$$\begin{bmatrix} L_{xx}(\mathbf{x}, \sigma) & L_{xy}(\mathbf{x}, \sigma) \\ L_{xy}(\mathbf{x}, \sigma) & L_{yy}(\mathbf{x}, \sigma) \end{bmatrix} \tag{1.6}$$

Dove L_{**} è la convoluzione di una derivata seconda di una Gaussiana con l'immagine I nel punto $\mathbf{x}$.

1.3.3 BRISK - Robust Independent Elementary Features

L'algoritmo *"BRISK"* (*Robust Independent Elementary Features*) è stato concepito nel 2011 da ricercatori dell'*Autonomous System Lab* dell'Università *ETH Zurich* [8]. Rispetto ai predecessori, il costo computazionale di questo algoritmo è molto ridotto. Tuttavia, ridotta è anche l'efficacia dell'algoritmo, ma dato l'enorme guadagno in termini di tempo, è generalmente considerato un prezzo accettabile. Si compone di tre moduli principali [9]:

- Individuazione dei punti chiave;
- Descrizione dei punti chiave;
- Corrispondenza dei descrittori.

Diversamente da *SIFT* e *SURF* che basano il riconoscimento di corrispondenza sulle regioni dell'immagini, questo algoritmo usa gli angoli delle immagini, che riconosce usando l'algoritmo AGAST (*Adaptive and Generic Accelerated Segment Test*) [10] e li filtra usando il punteggio restituito dall'algoritmo FAST (*Features from Accelerated Segment Test*) [11].
La costruzione del descrittore di questo algoritmo è data dal pattern di campionamento, che stabilisce N posizioni equidistanti in cerchi concentrici. Si definisce quindi un set A di coppie di punti dato da tutte le coppie di punti di campionamento:

$$A = \{(p_i, p_j) \in R^2 \times R^2 |\ i < B \wedge j < i, j \in N\} \tag{1.7}$$

Le coppie punti sono analizzate usando il rapporto che hanno in scala di grigi. Per analizzare il gradiente locale tra i due punti p_i e p_j, si prendono in considerazione i pixel smussati dai punti stessi annotandoli come $I(p_i, \sigma_i)$ e $I(p_j, \sigma_j)$.
L'analisi è data da:

$$g(p_i, p_j) = (p_i - p_j) \cdot \frac{I(p_i, \sigma_i) - I(p_j, \sigma_j)}{\|p_j - p_i\|} \tag{1.8}$$

Dove i e j sono ≥ 1 e $\leq N$.
Detta t la scala spaziale dei punti e σ_{max} e σ_{min} le soglie secondo cui definire due set S (breve distanza) e L (lunga distanza), abbiamo:

$$S = \{(P_i, P_j) \in A |\ \|P_i - P_j\| < \sigma_{max}\} \subseteq A \tag{1.9}$$

$$L = \{(P_i, P_j) \in A |\ \|P_i - P_j\| < \sigma_{min}\} \subseteq A \tag{1.10}$$

La direzione complessiva del gradiente dei punti d'interesse viene stimato dal set *L* come:

$$g = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \frac{1}{l} \cdot \sum_{(p_i, p_j) \in L} g(p_i, p_j) \tag{1.11}$$

L'invarianza rispetto alla scala e alla rotazione è gestita tramite la rotazione dell'angolo θ attorno al punto d'interesse k.

$$\theta = actan2(g_y, g_x) \tag{1.12}$$

Per quanto riguarda invece l'intensità sulla breve distanza, definita nel set S, il descrittore viene prodotto da:

$$b = \begin{cases} 1 & I(p_j^\theta, \sigma_j) > I(p_i^\theta, \sigma_i) \\ 0 & \text{altrimenti} \end{cases} \quad (P_i^\theta, P_j^\theta) \in S \tag{1.13}$$

p_i^θ rappresenta il punto p_i che ruota attorno al punto k ruotando θ.
La fase della corrispondenza delle caratteristiche, il cosiddetto *Feature Matching*, si realizza tramite il confronto tra le somiglianze dei descrittori di due punti caratteristici.

Il Feature Matching è effettuato usando la distanza di Hamming per i descrittori binari, differentemente da *SIFT* e *SURF* che utilizzano le norme L1 o L2 per descrittori a base di stringhe.

Per misurare la distanza in maniera efficace vengono usate operazioni bit a bit XOR. Ipotizzando di avere due descrittori *X* e *Y*, allora:

$$X = \chi_1 \chi_2 \cdots \chi_i \cdots \chi_N \tag{1.14}$$

$$Y = \gamma_1 \gamma_2 \cdots \gamma_i \cdots \gamma_N \tag{1.15}$$

dove il valore χ_i e γi può essere o 0 o 1.

L'equazione che esprime la distanza di Hamming è:

$$HD(X, Y) = \sum_{i=1}^{N} \chi_i \oplus \gamma i = \sum_{i=1}^{n} b(\chi_i, \gamma i) \tag{1.16}$$

dove $\oplus$ è lo XOR, e b$(\chi_i, \gamma i)$ è l'espressione del i-esimo bit di X e Y:

$$b(x, y) = \begin{cases} 1 & x \neq y \\ 0 & x = y \end{cases} \tag{1.17}$$

In base al valore restituito dalla funzione HD si ha che più è alto, minore è il grado di corrispondenza tra descrittori.

1.3.4 FREAK - Fast Retina Keypoint

L'algoritmo *"FREAK"* (*Fast Retina Keypoint*) è stato proposto nel 2012 [12]. È stato poi incluso in librerie di "Computer Vision" come *"OpenCV"*, come alternativa agli algoritmi precedentemente descritti. Il fondamento della proposta è basato sulla topologia della retina umana, la quale compie in autonomia una codifica spaziale. L'algoritmo inizia usando una "griglia a campionamento retinico" [12], la quale si sviluppa in maniera circolare (Figura 1.5), ha la particolarità di avere vicino al centro una densità maggiore di punti, e minore più ci si allontana da esso.

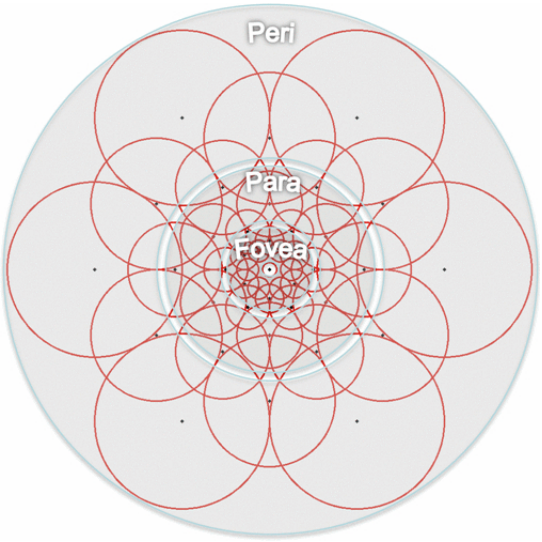

Figura 1.5: Illustrazione del pattern di campionamento FREAK simile alla distribuzione delle cellule gangliari retiniche con i corrispondenti campi ricettivi. Ogni cerchio rappresenta un campo ricettivo in cui l'immagine viene levigata con il corrispondente kernel Gaussiano [12]

La sovrapposizione dei cerchi permette usare meno campi recettivi, poiché viene presa in considerazione come informazione da codificare, esattamente come per l'occhio umano che utilizza una strategia simile. Il descrittore binario, che per brevità chiameremo F, è costruito imponendo una soglia tra le coppie di campi recettivi e i loro Kernel Gaussiani (1.18) corrispondenti.

$$K(u) = \frac{1}{\sqrt{2\pi}} e^{-\frac{1}{2}u^2} \tag{1.18}$$

Quindi ogni stringa binaria è composta da DoG rappresentati da un bit (1.19).

$$F = \sum_{0 \le a < N} 2^a T(P_a) \tag{1.19}$$

dove P_a è la coppia di campi recettivi e N la grandezza del descrittore desiderata e:

$$T(P_a) = \begin{cases} 1 & \text{se } (I(P_a^{r_1}) - I(P_a^{r_2}) > 0 \\ 0 & \text{altrimenti} \end{cases} \tag{1.20}$$

dove I è l'intensità. Questa serie di operazioni porta ad avere un descrittore di una certa grandezza e per raffinarlo si seguono i seguenti passi:

1. Dai punti corrispondenti estratti si crea una matrice D, in cui ogni riga rappresenta un punto chiave dato da tutte le possibili coppie. Si usano quindi 43 campi recettivi, questo porta ad avere approssimativamente 1000 coppie
2. Per ogni colonna si calcola la media.
3. Si ordinano le colonne rispetto alla varianza più alta.
4. Prendiamo le colonne la cui media è di 0.5 e aggiungiamo a queste le rimanenti colonne che hanno una bassa correlazione con esse.

Questo passaggio da grossolano a fine definisce la struttura del descrittore. Per mimare i movimenti individuali discontinui dell'occhio, si analizza in diversi step il descrittore. Si inizia l'analisi prendendo i primi 16 bytes che rappresentano l'informazione grossolana, e se la distanza è minore di una soglia si va avanti nella ricerca verso informazioni più fini. Facendo così si accelera il processo di matching e si eliminano più del 90% dei candidati. Per calcolare la rotazione di un punto chiave, si sommano i gradienti locali stimati come nell'algoritmo BRISK. Si prende un set G di tutte le coppie usate per calcolare i gradienti locali:

$$O = \frac{1}{M} \sum_{P_o \in G} (I(P_o^{r_1}) - I(P_o^{r_2})) \frac{P_o^{r_1} - P_o^{r_2}}{\|P_o^{r_1} - P_o^{r_2}\|} \tag{1.21}$$

dove M è il numero delle coppie dentro G a e $P_o^{r_i}$ è il vettore delle coordinate spaziali in due dimensioni del centro del campo ricettivo.

2 Plugins per Fiji

Cosa è *"Fiji"*? Come è nato? Quali sono le differenze con *ImageJ* e come si sviluppa un plugin per l'applicativo preso in esame? Queste sono le principali domande a cui daremo risposta in questo capitolo.

2.1 ImageJ

"ImageJ" è stato sviluppato nel 1997 dal *NIH* (*"National Institutes of Health"*). È un software open-source sostenuto da una vasta community di sviluppatori e ricercatori e oggi ed uno dei software più utilizzati da medici e biologi a fini scientifici. È conosciuto all'interno della lista di distribuzioni ad esso collegate come *"ImageJ 1.x"* o, per brevità IJ1. L'ultima versione risale al 31 marzo 2022 (*1.53q*), ma al giorno d'oggi ancora attivo e ampiamente scaricato. È considerato un tool capace di adattarsi alla maggior parte dei problemi legati all'analisi di immagini, ed è per questo usato sia nel campo della microscopia che ad esempio in quello della scienza dei materiali. La libreria che lo compone è accessibile via API, permettendo a sviluppatori del settore di riusarne e estenderne le funzionalità attraverso la creazione di *Plugin*, e di registrare *Macro* automatizzando processi interni attraverso linguaggi come Java, Beanshell, Javascript e una DSL (*Domain Specific Language*) per IJ1. L'utilizzo delle Macro permette ad utenti non Computer Scientists di non dover seguire un percorso di apprendimento particolare per l'automatizzazione di specifici workflow [13].

2.2 Differenze tra ImageJ2 e ImageJ 1.x

La seguente tabella è proveniente dalla fonte [14] e non è stata tradotta in lingua italiana per evitare incomprensioni.

	ImageJ 1.x	ImageJ2
Image types	uint8, uint16, float32, rgb	bit, uint2, uint4, uint8, uint12, uint16, uint32, uint64, uint128, int8, int16, int32, int64, float32, float64, cfloat32, cfloat64, bigint, bigdec, argb, <your-image-type-here>
Plugin types	PlugIn, PlugInFilter, PlugInTool	Command, Op, Tool, IOPlugin, Service, Converter, Codec, Format, DataHandle, TextFormat, ScriptLanguage, Display, UserInterface, Platform, App, Gateway, ModulePreprocessor, ModulePostprocessor, CodeRunner, ..., <your-plugin-type-here>
Dimensions	5D—XYZCT	N-dimensional—X, Y, Z, time, channel, emission spectra, lifetime, cell polarity, <your-dimension-here>
Image formats	HandleExtraFileTypes	SCIFIO—including Bio-Formats plugin; <your-format-here>
Parameters	GenericDialog	SciJava module @ parameter syntax—callable from ImageJ, KNIME, OMERO, CellProfiler, ..., <your-tool-here>
Script languages	IJ1 macro, JavaScript, BeanShell, Java	BeanShell, Clojure (Lisp), Groovy, Java, JavaScript, JRuby, Jython (Python), Renjin (R), Scala, <your-language-here>
User interface	AWT	Legacy, Swing, AWT, Apache Pivot, Eclipse SWT, JavaFX, KNIME, CellProfiler, OMERO, <your-ui-here>
Distribution	Download, install and update manually	Hundreds of update sites; receive updates automatically (when you want!)

2.3 ImageJ2

Il progetto *"ImageJ2"* è nato nel 2009 ed è stato costruito su *SciJava Common* che ne rappresenta il core. È il vero successore di *ImageJ* 1.x. La libreria è stata infatti riscritta, ripensata e totalmente ridisegnata, facilitando l'estensibilità di tutti i tipi di dato. Segue i dettami dei principi *SOLID* separando i compiti, disaccoppiando l'UI dal dominio applicativo, enfatizzando così la possibile integrazione con qualsiasi applicazione e/o libreria e portando il progetto ad essere interoperabile. Si può infatti intendere il progetto come un framework

condiviso per l'elaborazione d'immagini attraverso il quale si può scrivere una sola volta il codice per eseguirlo su diversi applicativi come *KNIME*, *CellProfiler*, *OMERO* e *Icy*. La compatibilità verso la prima versione è una delle principali feature di cui dispone e ciò permette una facile migrazione agli sviluppatori tramite la libreria *ImageJ Legacy*. È provvisto anche della libreria *ImageJ Updater* per fare l'aggiornamento di ogni singolo plugin della libreria *Imagej Ops* che comprende una serie di algoritmi di elaborazione immagini riusabili e di *ImageJ Common* che usa *"ImgLib2"* ed il core del modello dei dati delle immagini. Infine per la maggior parte delle operazioni di I/O fa uso della libreria *SCIFIO* (*"SCientific Image Format Input and Output"*) [15] [16] [17] [18].

2.4 Fiji

È l'acronimo per *Fiji is Just ImageJ*. L'architettura nasconde al suo interno la libreria *ImageJ2* e fornisce un'estensiva lista di plugin forniti di default. L'obiettivo principale di Fiji è migliorare l'esperienza utente dei ricercatori che non hanno conoscenze legate alla programmazione. I plugin forniti sono nella quasi totalità provvisti di una documentazione estensiva, dove si è guidati sia a livello generale con una prima panoramica delle funzionalità, sia a livello specifico in tutorial e dataset scaricabili. Il successo e la popolarità di *Fiji* è dato anche dalla sua capacità di integrarsi per operazioni e compiti troppo onerosi per l'applicazione con applicativi terzi. *Fiji* si interfaccia inoltre con MATLAB e *ITK*, e attraverso *ImgLib* e *ImageJ2* e permette l'adozione di ogni piattaforma scritta in Java [19].

2.5 Sviluppo di un plugin

2.5.1 Primi passi

Prima di tutto si deve creare un progetto Java con un IDE a piacimento (*"IntellJ IDEA"* o *"Eclipse"* ad esempio) e creare un file pom.xml all'interno della radice del progetto. Per vedere un esempio pratico di questo osservare *UnibosDS4H/DS4H* oppure *imagej/example-imagej2-command*.

```
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
...
<modelVersion>4.0.0</modelVersion>
<groupId>DS4H-Image-Alignment</groupId>
<artifactId>DS4H-Image-Alignment</artifactId>
<version>1.0.4</version>
<parent>
    <groupId>org.scijava</groupId>
    <artifactId>pom-scijava</artifactId>
    <version>25.0.0</version>
    <relativePath />
</parent>
<repositories>
    <repository>
        <id>imagej.public</id>
        <url>http://maven.imagej.net/content/groups/public</url>
    </repository>
</repositories>
    <dependencies>
        <dependency>
            <groupId>net.imagej</groupId>
            <artifactId>imagej</artifactId>
        </dependency>
    </dependencies>
</project>
```

Codice 1: Esempio parziale di file pom.xml UnibosDS4H/DS4H su Github

2.5.2 Il codice sorgente

Si può scegliere di usare ImageJ2 o ImageJ 1.x, il quale viene tradotto automaticamente tramite un componente interno chiamato *"Java Assist"*. Il primo esempio (Codice 2) è quello consigliato al momento ed è la via preferibile, mentre il secondo esempio (Codice 3 e Codice 4) è la versione Legacy che si può usare senza temere ma non permette di sfruttare le ultime features.

```
package hello;

import org.scijava.command.Command;
import net.imagej.ImageJ;

@Plugin(type = Command.class)
public class HelloWorld implements Command {
    /**
        * Main method is actually used only for debugging
        **/
    public static void main(final String... args) {
        final ImageJ ij = new ImageJ();
        ij.launch(args);
        ij.command().run(HelloWorld.class, true);
    }
    @Override
    public void run() {
        // ...
    }
}
```

Codice 2: Esempio semplice di un plugin Fiji in cui si fa uso di ImageJ2

```java
package hello;

import ij.IJ;
import ij.plugin.PlugIn;

public class HelloWorld implements PlugIn {
    @Override
    public void run(String s) {
            // ...
    }
}
```

Codice 3: Esempio semplice di un plugin Fiji in cui viene utilizzato ImageJ 1.x

```
Plugins>Hello, "Hello World", hello.HelloWorld
```

Codice 4: Esempio di file plugins.config obbligatorio nel caso dell'uso di ImageJ 1.x

2.5.3 Transizione da ImageJ 1.x a ImageJ2

Per chi ha già esperienza con la prima versione, qui può trovare una cheat sheet utile per effettuare la transizione alla seconda Figura 2.1 e Figura 2.2.

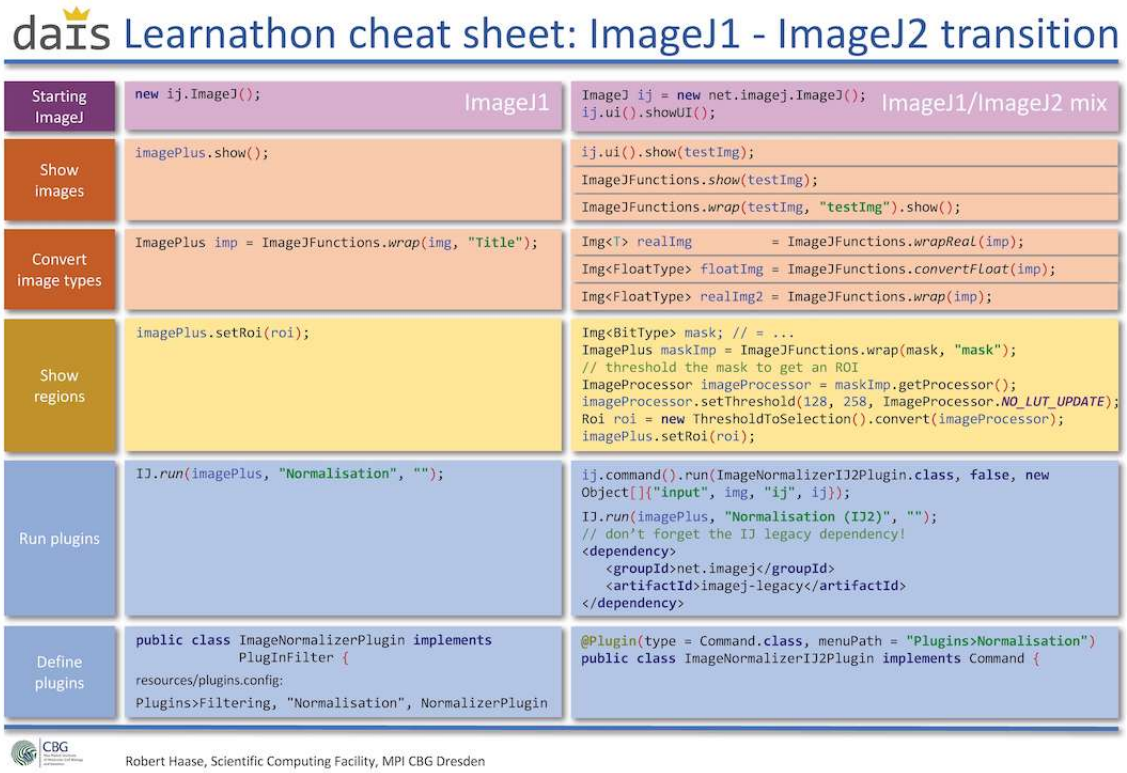

Figura 2.1: Pagina 1, di Robert Haase, Scientific Computing Facility, MPI CBG Dresden

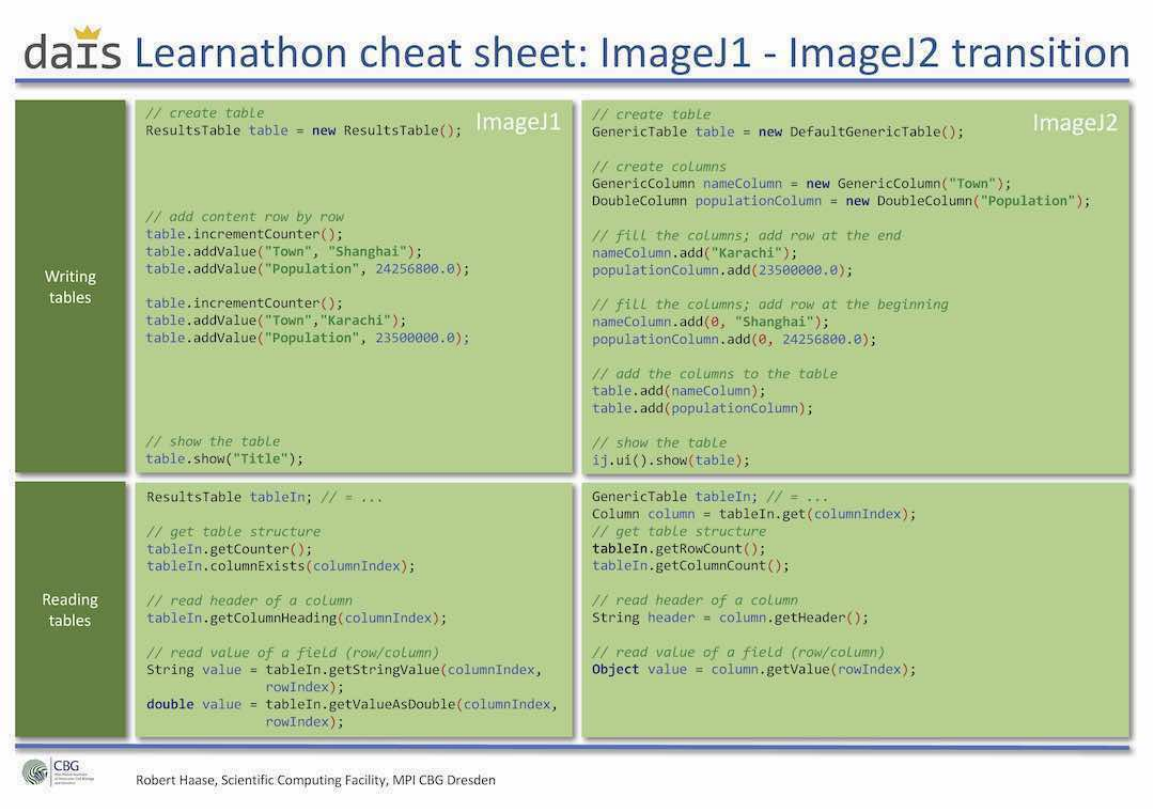

Figura 2.2: Pagina 2, di Robert Haase, Scientific Computing Facility, MPI CBG Dresden

2.5.4 Build

All'interno del file pom.xml dev'essere presente una parte di codice per eseguire il processo di creazione del file JAR, una di queste permette di includere agli assets Codice 5.

```
<build>
        <resources>
        <resource>
            <directory>src/main/assets</directory>
            <includes>
                <include>**/*</include>
            </includes>
        </resource>
    </resources>
</build>
```

Codice 5: pom.xml, sottosezione build

Per creare il file JAR nella giusta cartella occorre creare, se non già presente in questo percorso **HOME/.m2**, un file chiamato **settings.xml** come mostrato in Codice 6

```
<settings>
    <profiles>
        <profile>
            <id>imagej</id>
            <activation>
                <file>
                    <exists>${env.HOME}/Desktop/Fiji.app</exists>
                </file>
            </activation>
            <properties>
                <scijava.app.directory>${env.HOME}/Desktop/Fiji.app </scijava.app.directory>
            </properties>
        </profile>
    </profiles>
</settings>
```

Codice 6: settings.xml valido per Fiji

Come compilare un file Maven dipende dall'IDE di riferimento. In questo paragrafo spiegheremo come si fa con *IntellJ IDEA*, il quale fornisce tre modalità. La prima è tramite Run → Edit Configurations -> Maven, la quale non ha bisogno normalmente di specifici comandi per l'inizializzazione, e permette una volta eseguito di creare il file JAR. La seconda modalità è attraverso il plugin stesso, mostrato in Figura 2.3, dove l'UI permette agli utenti meno esperti di usufruire dei comandi e dei profili Maven senza avere conoscenze pregresse.

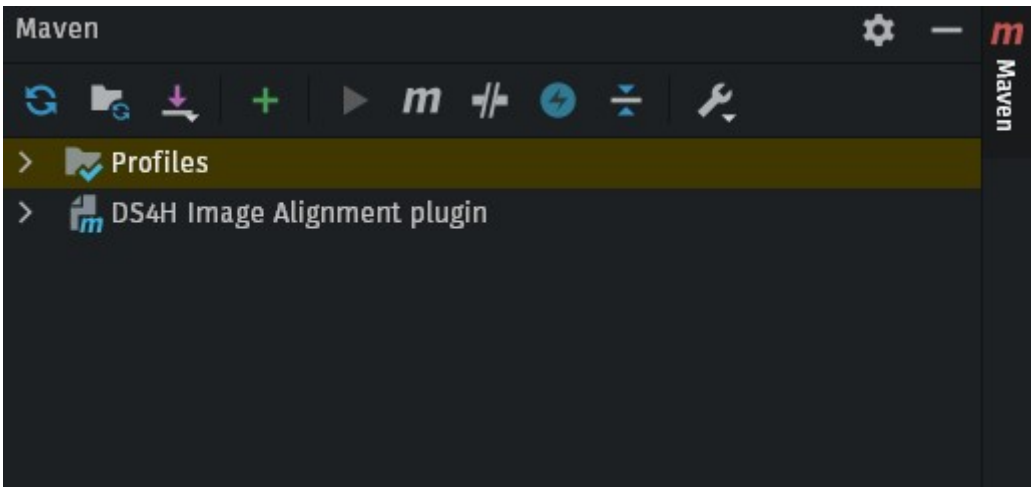

Figura 2.3: Esempio plugin Maven IntelJ IDEA

L'ultima modalità è quella di digitare i comandi specifici direttamente dal terminale, ad esempio Codice 7:

```
mvn clean & mvn compile & mvn install
```

Codice 7: Esempio per pulire, compilare e creare il file JAR

2.5.5 Service

A differenza di *ImageJ 1.x*, dove per accedere alla maggior parte delle sue funzionalità si usano dei metodi statici, per *ImageJ2* ogni parte ha la sua logica di business incapsulata in un servizio. Un servizio non è altro che una classe con una serie di responsabilità legate ad un solo argomento. *ImageJ2* dispone di tantissimi Service, tra questi possiamo ricordare:

- AppService - tiene traccia l'applicazione presente nel contesto.
- DisplayService - tiene traccia delle schermate disponibili, di quella attiva, e da l'abilità di crearne di nuove.
- EventService - gestisce gli eventi legati all'applicazione, ed è la base per gestire la comunicazione tra le diverse parti dell'applicativo.
- IOService - contiene gli strumenti per aprire e salvare file all'interno del contesto.
- MenuService - ha la struttura del menu dell'app che può essere costruita attraverso questo servizio.
- ModuleService - tiene traccia dei moduli disponibili, e fornisce l'infrastruttura per eseguirli.
- ObjectService - tiene traccia degli oggetti disponibili, inclusi i Datasets e i Displays.
- OptionsService - ha gli strumenti per gestire le impostazioni del programma.
- PlatformService - provvede hooks per estendere l'applicativo rispetto al sistema operativo in uso.
- PluginService - tiene traccia dei plugins, e fornisce gli strumenti per eseguirli.
- ScriptService - permette di eseguire gli script e macro.
- StatusService - tiene traccia dello stato delle operazioni in corso.
- ThreadService - gestisce multithreading, utile per eseguire più operazioni e task in parallelo.
- UIService - esegue le UI per interagire con ImageJ.
- DatasetService - ha gli strumenti per creare e gestire i dati delle immagini.
- ImageDisplayService - simile a DisplayService, ma per gli ImageDisplays.
- OverlayService - ha gli strumenti per creare e gestire gli overlays e le ROIs delle immagini.
- FormatService - servizio per gestire i formati delle immagini disponibili.

2.5.6 Consigli

La fase più importante per la creazione di software è la quella di progettazione del dominio e della business logic. È importante non pensare alle librerie che si useranno, poiché rappresentano una parte terza che dev'essere sempre possibile cambiare, ad esempio se si sceglie una differente libreria di UI durante la fase di sviluppo rispetto a quella già implementata, la modifica si deve poter fare senza dover cambiare parti del dominio. È buona pratica studiare il linguaggio usato, conoscere cosa si intende per OOP e cosa sono i principi SOLID.

3 Descrizione del DS4H Image Alignment Tool

Il capitolo corrente descrive la prima versione del plugin *DS4H Image Alignment*, progetto sviluppato da Stefano Belli nel 2019 come Tesi Magistrale in questo corso di studi.

3.0.1 Presentazione generale

Il plugin ha una GUI (textitgraphical user interface) organizzata in finestre modali. Le principali sono:

- Finestra principale - è la finestra incaricata di mostrare l'immagine corrente. Permette di aggiungere corners per l'allineamento manuale, e da accesso a tutte le altre finestre di dialogo del software.
- Finestra di preview - mostra una anteprima delle immagini riportando la lista delle coordinate dei corners già inseriti.
- Finestra per la cancellazione - permette di rimuovere una o più immagini precedentemente caricate.
- Finestra di co-registrazione - finestra che mostra il risultato della co-registrazione delle immagini tramite corners, permette di salvare il risultato in file TIFF o di riutilizzare le stesse immagini per un nuovo ciclo di co-registrazione.

A livello utente è da notare che ogni immagine presente nella sequenza ha legata a se un ROIManager, strumento presente nativamente all'interno di *ImageJ* e *Fiji*. Questo strumento permette di gestire i corner point, che sono punti d'interesse che l'utente stesso annota all'interno di ogni immagine per poi utilizzare le funzionalità di co-registrazione.

3.0.2 OME Data Model

Ogni immagine importata tramite finestra di dialogo (Figura 3.1), racchiude al suo interno delle informazioni chiamate metadati. Per poter codificare ed analizzare i metadati, il plugin sfrutta una libreria chiamata *"Bio-Formats"* sviluppata dall'*"Open Microscopy Environment"*, consorzio di università, laboratori di ricerca e aziende che lavorano in ambito bio-medicale

per uniformare formati e software open source. Uno dei loro prodotti più famosi è *"OMERO"*, software web-based di editing di immagini scientifiche, nonché il formato standard per l'archiviazione e consultazione di immagini microscopiche 2D chiamato *"OME Data Model"*. Lo standard in questione integra i metadati sopracitati, permettendone l'acquisizione, l'analisi e la trasformazione. È definito in uno schema XML, ed è liberamente consultabile, e permette di aggiungere annotazioni chiamate StructuredAnnotations. Il formato file più conosciuto e usato in ambito di imaging biomedicale dell'*OME Data Model* al momento è *OME-XML* ed il più recente *OME-TIFF* [20] [21] [22].

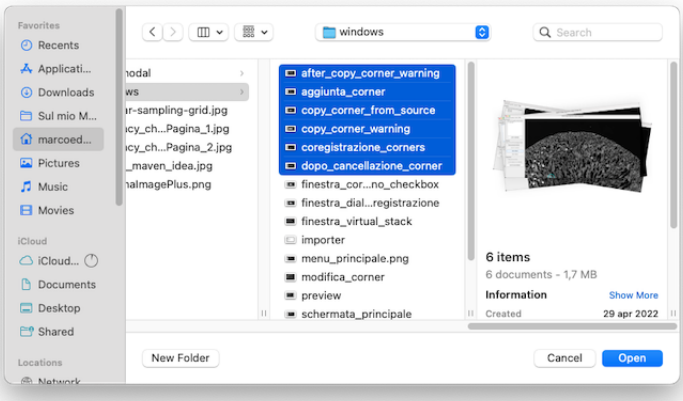

Figura 3.1: Finestra di importazione immagini, è possibile sceglierne più di una

3.0.3 Modalità d'uso dell'applicazione DS4H

Di seguito sono spiegate tutte le funzionalità disponibili all'utente. Le schermate presentate in questa sezione sono state esportate utilizzando un Macbook Pro M1 Pro.

Aggiunta Corner Point

Premendo il tasto **C** all'interno dei limiti dell'immagine corrente, un corner point verrà creato rispettivamente al punto in cui si è premuto il tasto. L'esempio in Figura 3.2 mostra il risultato dell'operazione, dove sono stati creati tre volte i corner point, e da cui si può notare l'ordine di creazione grazie alla numerazione automatica.

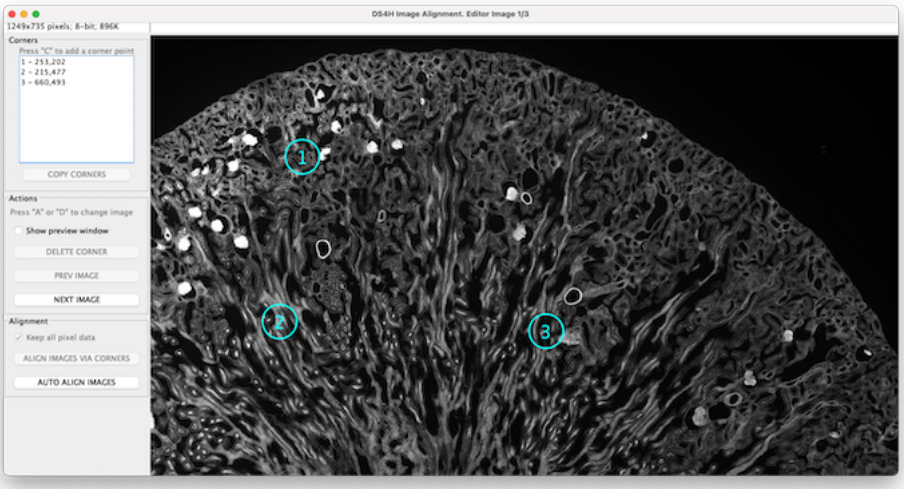

Figura 3.2: Schermata principale in cui sono stati aggiunti dei corner points

Modifica Corner Point

Ogni corner point è modificabile trascinandolo tramite cursore come mostrato in Figura 3.3.

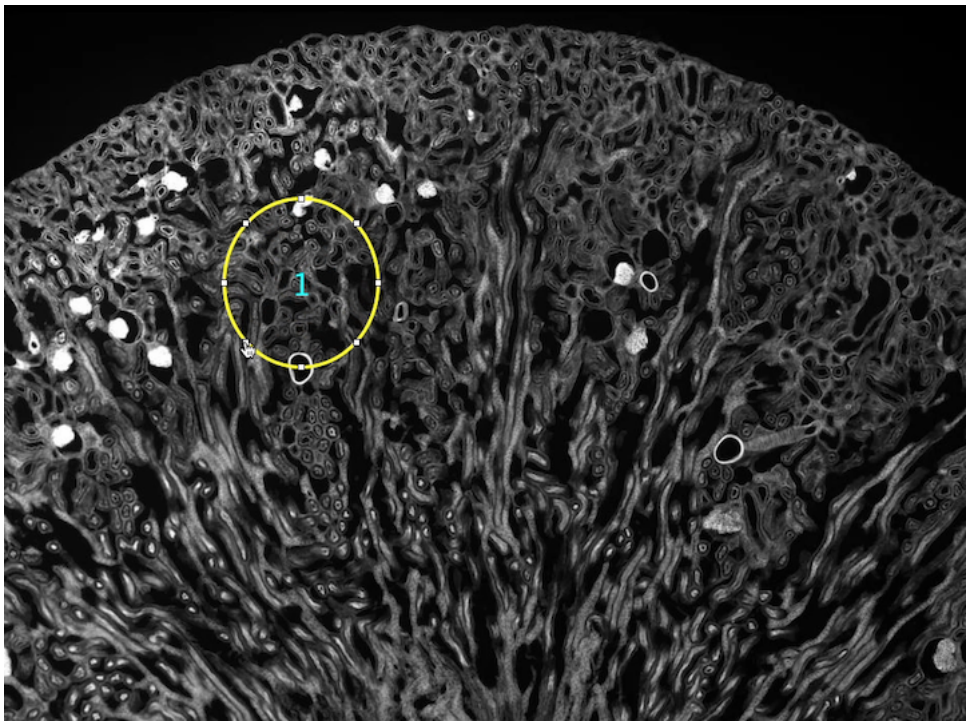

Figura 3.3: Immagine corrente in cui sta venendo modificato un corner

Rimozione Corner Point

Ogni corner point una volta creato è visibile e selezionabile anche attraverso una lista nella parte sinistra della finestra principale, ed una volta selezionato tramite cursore si può cancellare premendo il bottone Figura 3.5 (“Delete Corner”). Il risultato della rimozione di un corner è mostrato in Figura 3.4

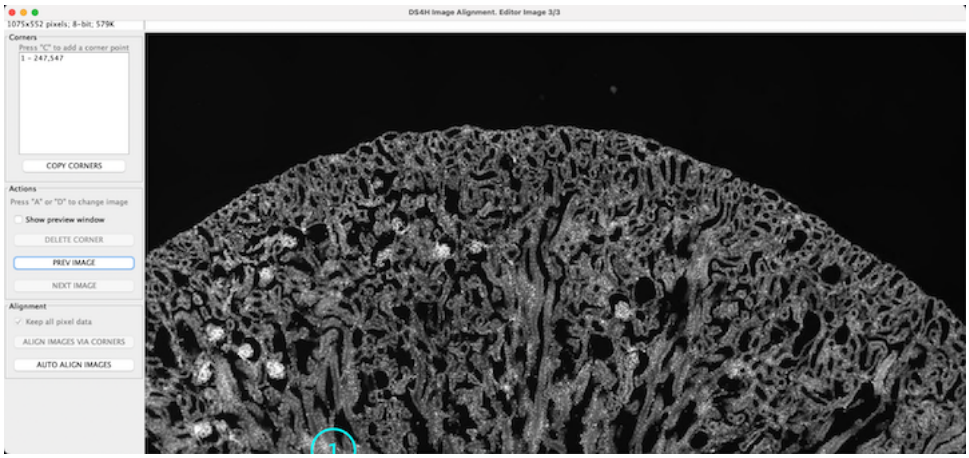

Figura 3.4: Risultato a seguito della rimozione di un corner

Cambia Image Corrente

I bottoni "Next Image" e "Prev Image" in Figura 3.5 permettono di navigare tra le immagini importate.

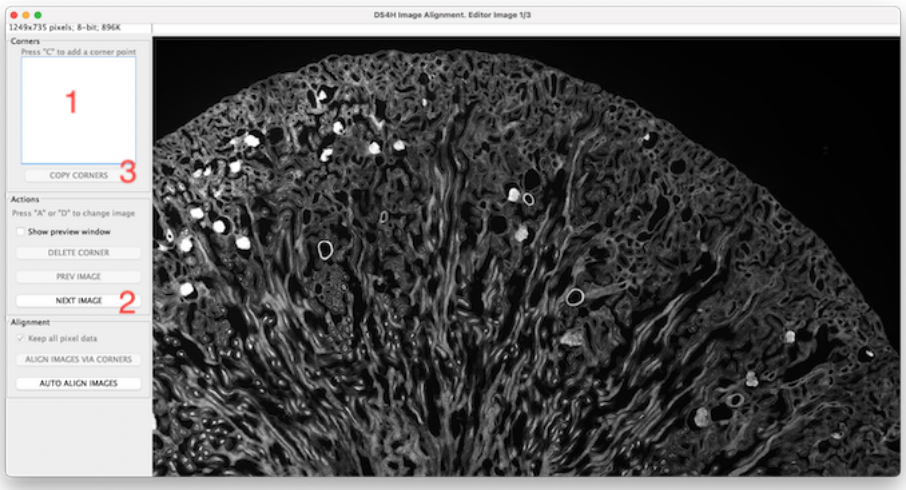

Figura 3.5: Schermata principale i cui numeri definiscono: 1 Rimozione Corner, 2 Cambia Immagine, 3 Copia Corner

Copia Corner Points

Per facilitare l'apposizione dei corners in tutte le immagini, l'applicazione è fornita di un bottone apposito visibile in Figura 3.5 che se premuto mostra una finestra di dialogo, come quello mostrato in Figura 3.6. Da questa finestra è possibile scegliere l'immagine da cui importare i corner points, e nel caso in cui i corner points importati risultino fuori dai limiti delle immagini il sistema notifica l'utente di ciò, dando la possibilità di cancellare anche dall'immagine sorgente i suddetti punti Figura 3.7.

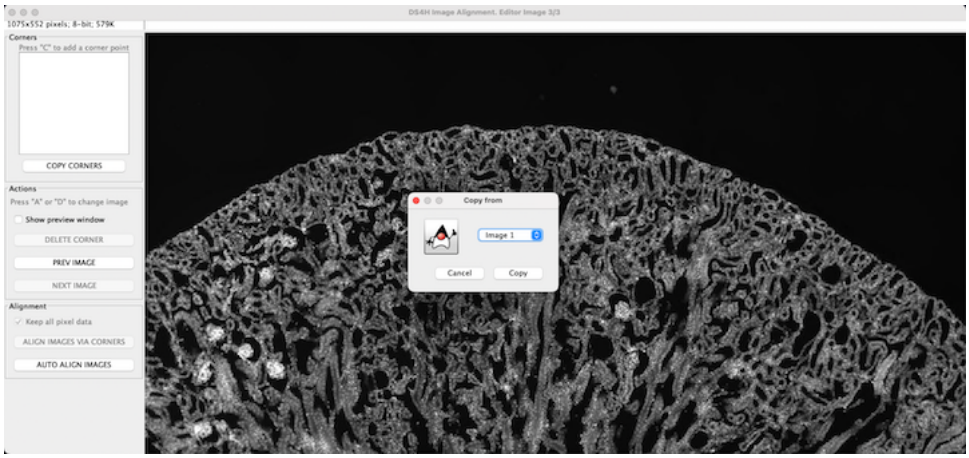

Figura 3.6: Finestra di dialogo da cui si può scegliere l'immagine sorgente per eseguire la copiatura

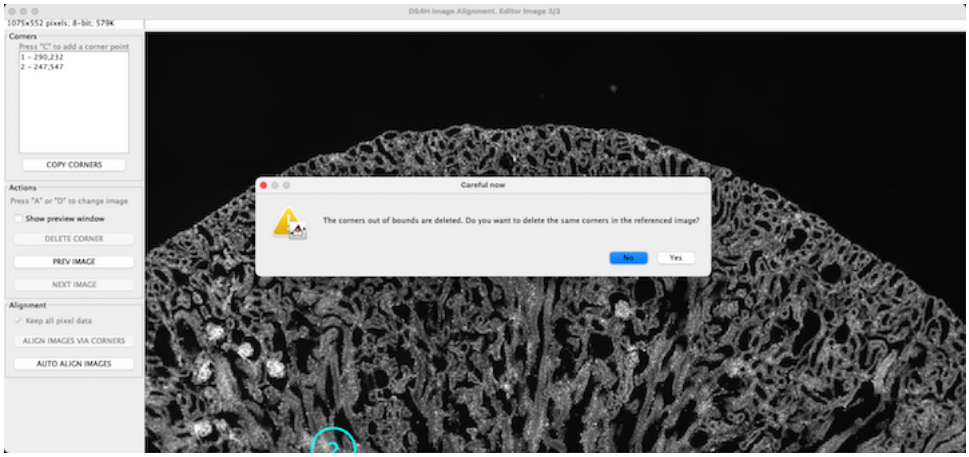

Figura 3.7: Finestra di warning che notifica l'utente la presenza di corners fuori dai limiti dell'immagine

Aggiunge e Rimuove File

Dalla finestra principale, accedendo al menu "File" (Figura 3.8) si possono eseguire sia l'importazione di una nuova immagine da aggiungere alla presente lista di immagini, sia rimuovere un immagine a scelta.

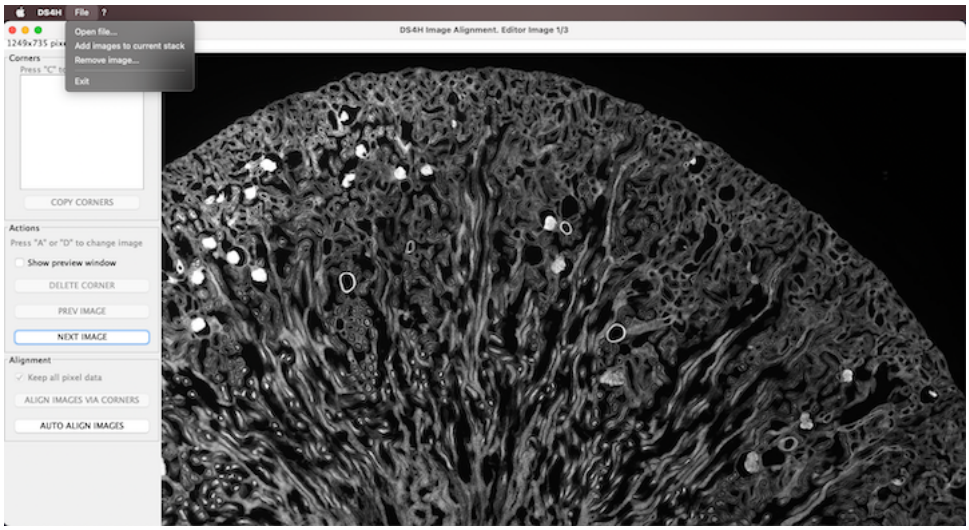

Figura 3.8: Schermata principale in cui sono stati aggiunti dei corner points

Selezione singola/multipla

Come precedentemente citato, vi è una sezione a sinistra in cui sono presenti in lista tutti i corner presenti, essi sono selezionabili sia in forma singola (Figura 3.9) che multipla (Figura 3.10). Tutti i corner points selezionati, si possono spostare insieme, se durante questa operazione dei corners finiscono fuori dai limiti dell'immagine corrente, si mostra una finestra di warning che avverte la prossima cancellazione degli stessi come in Figura 3.11.

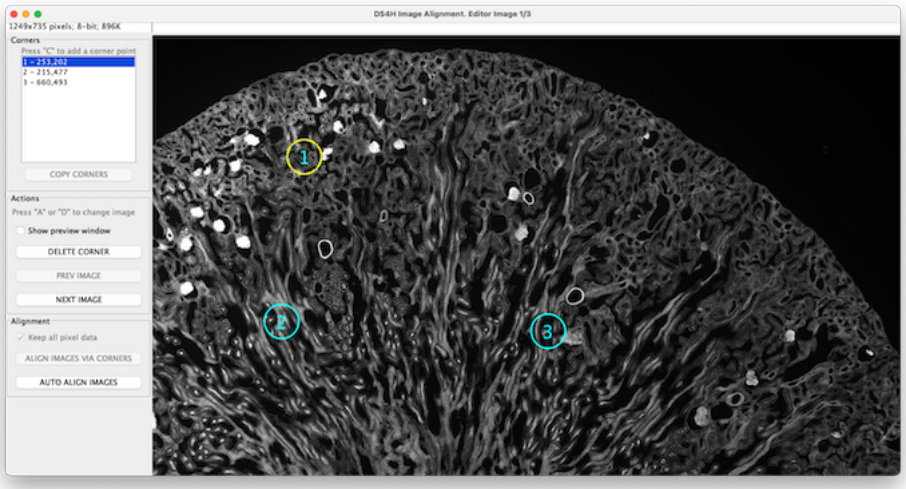

Figura 3.9: Esempio selezione singola

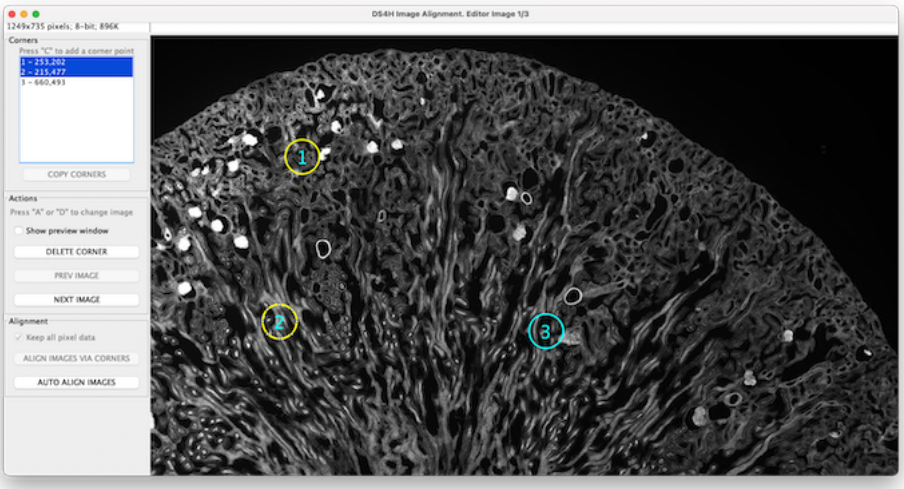

Figura 3.10: Esempio selezione multipla

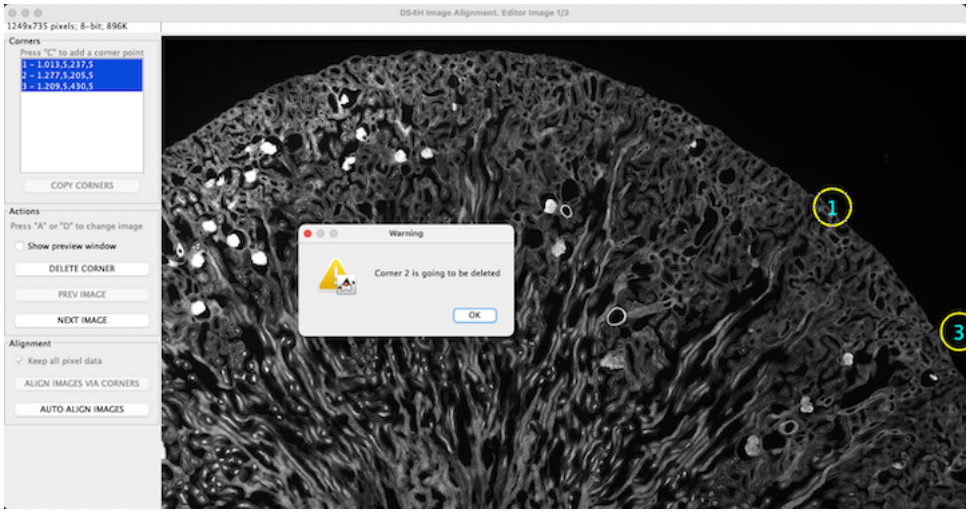

Figura 3.11: Esempio trascinamento dopo aver selezionato più corners, con finestra di warning

Esecuzione Co-registrazione

Premendo il bottone “Align Images via Corners” si può sfruttare una delle due funzionalità principali dell’applicazione. Pre-requisito per poterne avviare l’esecuzione è definire almeno tre corner points all’interno di ogni immagine presente. Come mostrato in Figura 3.12, viene poi richiesto quale tipo di trasformazione si vuole applicare tra le immagini. Nel caso in cui non venga scelta nessuna delle opzioni presenti (“Projective” o “Rigid”) il sistema applica di default il modello affine. Il modello affine prevede traslazioni, rotazioni e cambi di scala, mentre il modello rigido solo traslazioni, e quello proiettivo anche deformazioni. Qualunque sia il risultato, esso viene mostrato come in Figura 3.13, e le immagini registrate sono allineate in uno stack navigabile.

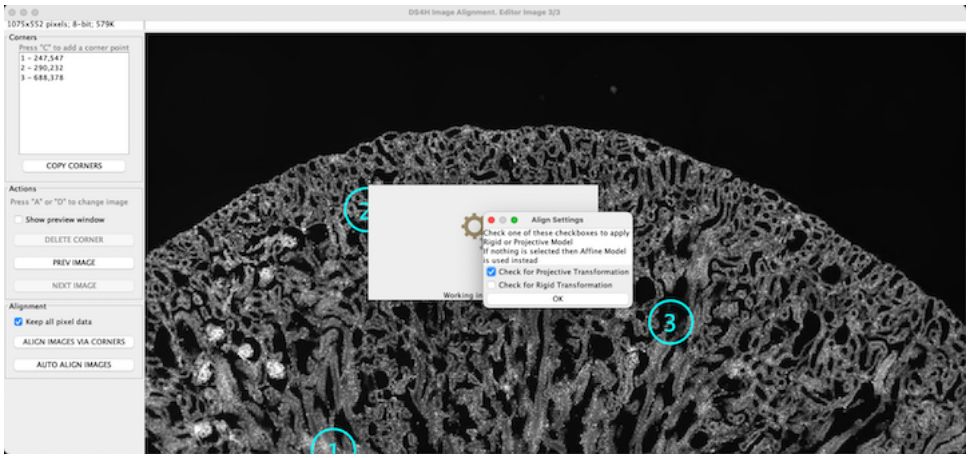

Figura 3.12: Finestra di co-registrazione per scegliere il modello di trasformazione

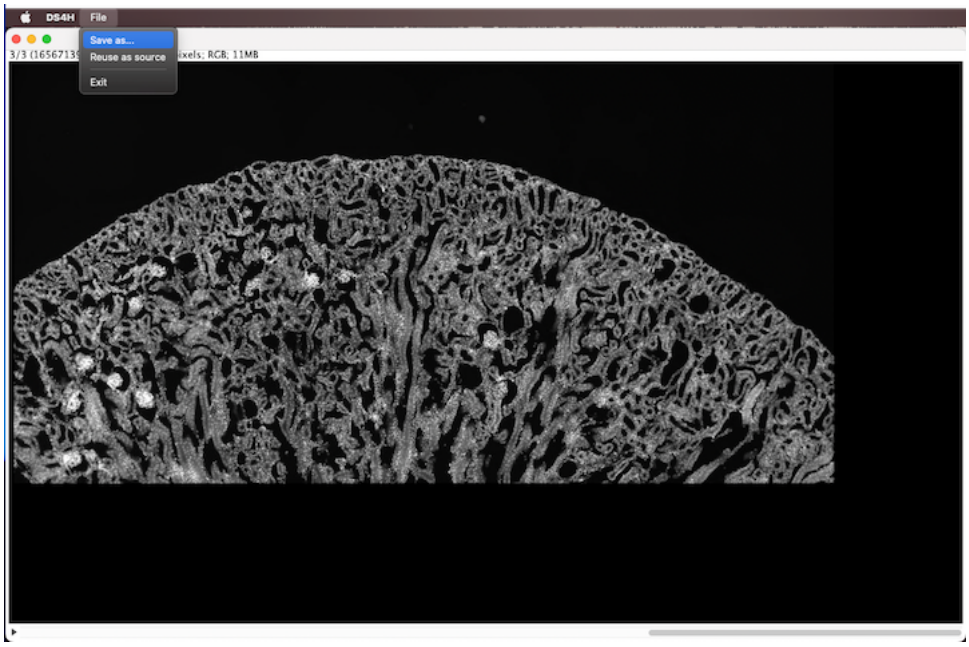

Figura 3.13: Finestra di co-registrazione in cui si mostra l'applicazione della trasformazione

Salvare Immagine risultante e Riutilizza Immagine

A seguito dell'operazione di co-registrazione, si può scegliere o di salvare in un unico file TIFF l'immagine risultante per un uso futuro, oppure ricaricare le immagini co-registrate per altre operazioni all'interno dell'applicazione. Entrambe le scelte sono visibili in Figura 3.13, e sono accessibili via menu della finestra. Nel caso in cui si scelga la seconda opzione il risultato è mostrato in Figura 3.14.

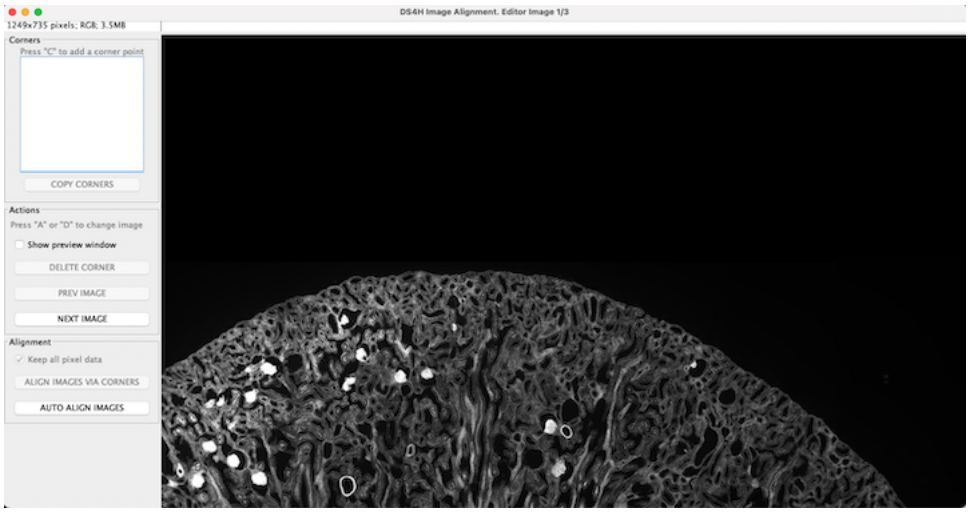

Figura 3.14: Finestra principale a seguito del riuso delle immagini co-registrate

Finestra di preview

La finestra di preview è utile per controllare la posizione dei corner point e facilitare l'apposizione nelle altre immagini presenti nella sequenza. Per accedervi basta premere il checkbox nella finestra principale che reca la scritta "Show preview window". La finestra in Figura 3.15 è navigabile e mostra i corner point, i quali non sono selezionabili, privi di numerazione.

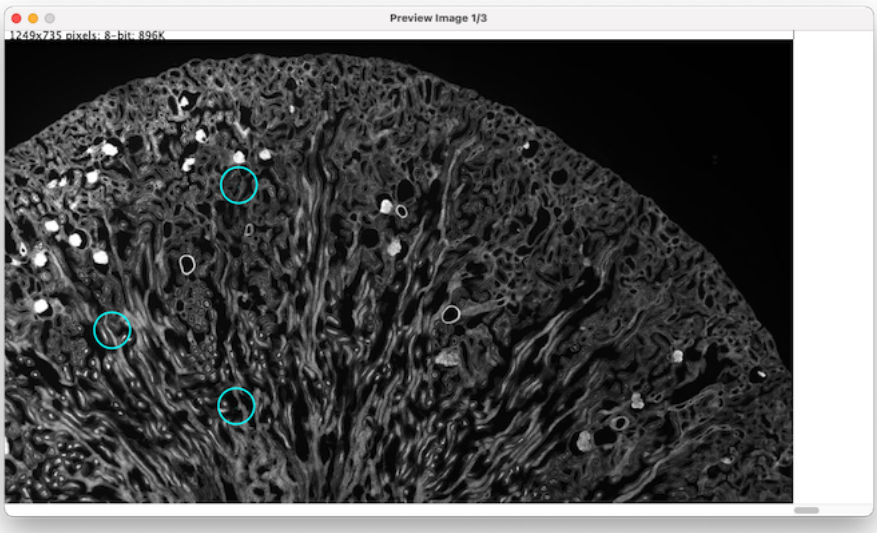

Figura 3.15: Finestra di preview

4 Descrizione dei competitors

4.1 Lista Tool Freely Available per Stitching Multimodale

Negli ultimi anni sono stati sviluppati vari tool per fare allineamento multi-modale di immagini. La maggior parte di essi richiede iterazione con l'utente che spesso deve definire manualmente i punti di riferimento nelle varie immagini. Tuttavia esistono anche alcune soluzioni completamente automatiche. Riportiamo qui di seguito una breve descrizione dei principali tool oggi disponibili.

4.1.1 Elastic Alignment and Montage

"Elastic Alignment and Montage" sono due plugin incorporati a loro volta nel software *"TrackEM2"*. Vengono sfruttati quando si hanno larghe serie di sezioni multi-tile attraverso il menu di allineamento. Il primo allinea serie di sezioni deformate, mentre il secondo crea mosaici da immagini sovrapposte le quali hanno una deformazione non lineare. Le immagini sono deformate in modo da ottenere delle corrispondenti sovrapposizioni ottime. Ogni singola deformazione è calcolata in modo da il montaggio finale sia deformato al minimo. Entrambi i plugin lavorano esclusivamente con stacks di immagini.

4.1.2 TrackEM2 Folder Watcher

TrackEM2 lavora sia con immagini 2D sia 3D. Permette di estrarre misurazioni e creare allineamenti multi-modali identificando manualmente le aree di sovrapposizione per ogni immagine appartenente allo stack. È possibile anche ordinare le stesse sezioni create raggruppandole in alberi gerarchici, ed effettuare annotazioni testuali. Sfrutta sia i plugin citati in precedenza per l'allineamento sia *SIFT* [23].

4.1.3 Moving Least Squares

"Moving Least Squares" sfrutta il metodo chiamato spostamento dei minimi quadrati. Le tecniche implementate per trasformazioni può essere utile solo deformare un'immagine singola dati una serie di punti di riferimento. Non si occupa di allineamento tra immagini. Le trasformazioni, in contrapposizione a *"bUnwarpJ"* che utilizza deformazioni elastiche, sono da intendere come rigide. Gli autori del plugin intendevano dimostrare che fosse

possibile creare deformazioni usando trasformazioni geometricamente similari, cioè usando trasformazioni rigide insieme a ridimensionamenti isometrici, evitando la minimizzazione non-lineare [24].

4.1.4 Linear Stack Alignment with SIFT

"Linear Stack Alignment with SIFT" sfrutta *SIFT* e *RANSAC* (*Random Sample Consensus*) per identificare automaticamente punti di riferimento tra diverse immagini e allineare le stesse senza richiedere l'intervento dell'utente. Viene utilizzata una trasformazione rigida per il secondo di due tagli che mappa le corrispondenze del secondo in maniera ottima a quelle del primo.

4.1.5 Register Virtual Stack Slices

"Register Virtual Stack Slices" prende una sequenza di immagini in input e restituisce in output l'allineamento finale registrato in un'unica immagine di riferimento. L'immagine identificata come riferimento intatta non è modificabile. È scale invariant poiché utilizza modelli estratti automaticamente via *SIFT*.

4.1.6 Image Stitching

"Image Stitching" richiede come input una griglia quadrata di immagini. Utilizza il Teorema di Fourier per calcolare tutte le possibili traslazioni tra tutte le immagini messe in input, in una volta. In output fornisce la miglior sovrapposizione in termini di misure di cross correlazione. È multi-modale poiché è capace di allineare un numero arbitrario di canali di natura differente. Per rimuove differenze di luminosità tra i bordi delle porzioni della griglia applica una correzione d'intensità non lineare. Per fare stitching tra due immagini permette di usare il *"Pairwise Stitching"*, dove si forniscono immagini in cui si sono definiti dei punti d'interesse e per calcolare le matrici di correlazione di fase. Con la modalità *"Grid/Collection Stitching"* si sistemano le immagini in griglia rispetto a tutte le possibili orientazioni (*e.g.* riga per riga, colonna per colonna).

4.1.7 BigStitcher

"BigStitcher" è il successore di *Image Stitching*, e mantiene le stesse funzionalità. È size invariant, può processare e mostrare immagini indipendentemente dalla loro grandezza. Al momento un'ottimizzazione globale è capace di allineare dataset connessi in modo sparso, dove il contenuto dell'immagine è separata da aree dove lo sfondo è quasi sempre costante. È un software ancora in versione di beta test, quindi non è da considerarsi completo. Tra le altre feature che fornisce vi sono:

- la registrazione multi-view dove si possono importare immagini provenienti il cui orientamento è diverso;

- l'elaborazione dei dati come la fusione o la deconvoluzione delle immagini allineate, anche per risoluzioni ridotte dell'immagine prodotta sia per selezioni di parte di essa;

- scelta della miglior direzione della luce per ogni immagine;

- supporto di griglie non regolari a cui ad esempio mancano delle parti.

4.1.8 MIST

"MIST" è l'acronimo di *"Microscopy Image Stitching Tool"*, un algoritmo di stitching per collezioni di griglie di immagini 2D. Gli autori hanno sviluppato un metodo per ottimizzare il processo di *Phase-Correlation* usando l'approccio della Trasformata di Fourier. Ottimizza il calcolo di tutte le traslazioni usando l'algoritmo *"Hill Climbing"* limitando la ripetibilità di questa fase a quattro volte. Questo metodo è valido in immagini sparse di grandi dimensioni con set di dati di riferimento, generate per calcolare gli errori di cucitura sul mosaico assemblato. Il software in questione è usufruibile solo per *ImageJ/Fiji* a 64 bit [25].

4.1.9 MicroMos

"MicroMos" è un software open-source per costruire automaticamente un mosaico di immagini attaccando (i.e. stitching) immagini parzialmente sovrapposte. Come prerequisito per le immagini fornite in input vi è il bisogno che siano statiche e indeformabili. Nella fase di stitching vengono applicati algoritmi anche per correggere l'effetto di vignettatura, cioè per correggere la riduzione di intensità o saturazione dell'immagine che può presentarsi nei contorni. Può essere usato per registrazione multi-modale ma l'utente deve definire a mano la posizione di sovrapposizione.

4.1.10 Correlia

"Correlia" è un software open-source. Ha un'interfaccia facile da usare, e si occupa sia di registrazione di immagini che presentano trasformazioni lineari, ovvero trasformazioni rigide come la traslazione, la rotazione e la scala, sia non lineari come le deformazioni elastiche. Sfrutta un modello di trasformazione basato sulle cubiche B-Spline, usando il plugin *bUnwarpj* di *ImageJ/Fiji* per quanto riguarda le deformazioni, mentre per le trasformazioni rigide usa l'interpolazione bilineare. È un software multi-modale, e fa uso degli algoritmi di warping per migliorare la registrazione di immagini che presentano colorazioni diverse [26].

4.1.11 BigWarp

"BigWarp" è uno strumento per l'allineamento di immagini multimodali in 2D e in 3D basato su punti di riferimento. La sua fase di registrazione è basata sull'uso di un'immagine source che funge da template e una target. Manualmente l'utente applica trasformazioni affini ed elastiche per adattare il target al source. Per visualizzare le immagini utilizza il plugin *"BigDataViewer"* e un'implementazione delle *Thin Plate Spline* per gestire le deformazioni. Permette di gestire immagini molto pesanti, e di trasformare punti i punti di riferimento rispetto ad un target, usando un file csv [27].

4.1.12 ec-CLEM

"ec-CLEM" è un plugin di *"Icy"*. Concorrente di *ImageJ*, utile per fare registrazione multi-modale. È un plugin multi-modale, multidimensionale poiché gestisce anche immagini in tre dimensioni permettendo l'allineamento di immagini in maniera rigida o deformata. Si serve di punti d'interesse manualmente applicabili, ma ha anche una funzione chiamata Autofinder che si occupa della registrazione automatica. Autofinder può usare per la registrazione automatica maschere di segmentazione oppure punti di riferimento. Il plugin vuole risolvere il problema di assicurarsi un alto grado di accuratezza senza curarsi dell'origine di acquisizione dell'immagine, sia essa stata acquisita con microscopi elettronici o con microscopi ottici. Da la possibilità di effettuare registrazioni rigide o elastiche: per quanto riguarda le seconde, il requisito fondamentale è che sia stata effettuata almeno una registrazione rigida iniziale [28].

4.1.13 Elastix

"Elastix" è un plugin di *Fiji*. La funzionalità principale registrazione di immagini mediche basate sull'intensità, sia in 2D che in 3D. È anch'esso un plugin multi-modale e multidimensionale, e permette l'allineamento sfruttando il modello traslativo, affine e proiettivo. *"Elastix"* ha due funzioni specifiche per l'allineamento automatico. La prima interessa le immagini in tre dimensioni ed è basata sulla trasformazione Euleriana tra punti di riferimento per identificare le deformazioni da attuare. La seconda può essere utilizzata in immagini a due dimensioni e gestisce le deformazioni tramite algoritmo noto con il nome di "BSpline". L'allineamento avviene tra due immagini una che rimane fissa e una a cui viene applicata la trasformazione. Il plugin è utilizzabile sia in modalità headless eseguendo via terminale dei comandi, sia via GUI con una finestra di dialogo. Il risultato può essere direttamente salvato come file TIFF, in una cartella di destinazione, o si può scegliere di mostrare una finestra con l'immagine risultante [29].

4.2 Tabella comparativa plugins/tools

Nome	Hyper-Link	Dataset disponibile [Y/N]	Tutorial disponibile [Y/N]	Gestisce scale differenti [Y/N]	Gestisce size differenti [Y/N]	Registrazione Manuale [Y/N]	Registrazione Automatica [Y/N]	Registrazione Automatica Multimodale [Y/N]
Elastic Alignment and Montage	elastic-alignment-and-montage	N	Y	Y	Y	N	N	Y
TrackEM2 Folder Watcher	trakem2	Y	Y	Y	Y	Y	Y	Y
Align Image by line ROI	align-image-by-line-roi	N	Y	Y	N	N	Y	Y
Moving Least Squares	moving-least-squares	N	Y	N	N	N	Y	N
Linear Stack Alignment with SIFT	linear-stack-alignment-with-sift	N	N	N	N	N	Y	N
Register Virtual Stack Slices	register-virtual-stack-slices	N	Y	Y	Y	N	Y	Y
BigStitcher	bigstitcher	Y	Y	N	Y	Y	Y	Y
ImageJ Stitching	image-stitching	N	Y	N	Y	N	Y	Y
MIST	mist	Y	Y	N	N	N	Y	N
MicroMos	micromos	Y	Y	N	N	Y	Y	N
Correlia	correlia	Y	Y	Y	Y	Y	Y	Y
BigWarp	BigWarp	N	Y	Y	Y	Y	Y	N
ec-CLEM	ec-CLEM	Y	Y	Y	Y	Y	Y	Y
Elastix	Elastix	N	Y	Y	Y	N	Y	Y

4.3 Limiti

La ricerca dei competitors ha evidenziato la presenza di quattro competitors:

- *TrackEM2*;
- *Correlia*;
- *BigWarp*;
- *ec-Clem.*
- *Elastix.*

Per ognuno di questi software si definisco di seguito i limiti riscontrati.

TrackEM2

TrackEM2 è predisposto principalmente per analizzare immagini pesanti particolarmente e consente il settaggio di vari flags. Questo è già un primo indicatore della sua poca *user-friendliness* poiché richiede conoscenze informatiche dell'utente per essere sfruttato appieno. L'applicazione non nasce per la registrazione, ma per il *data mining* morfologico. Il suo utilizzo principale è la misurazione di volumi, superfici e lunghezze via punti di riferimento. L'utilizzo di *SIFT* come algoritmo di registrazione predispone il tool ad un ulteriore peso sotto il profilo del costo computazionale. L'allineamento via *SIFT*, seppur automatico, richiede più step attraverso finestre di dialogo per la configurazione. Questi elementi non permette un veloce utilizzo, e per quanto sia un ottimo software, risulta macchinoso.

Correlia

Correlia presenta un limite sostanziali: come per quasi tutti i competitors elencati, richiedere all'utente il tipo di trasformazioni da applicare durante la fase di allineamento automatico, rendendo il suo utilizzo particolarmente lento. Altri punti negativi di cui tenere in considerazione sono l'utilizzo di algoritmi datati e la dipendenza da *bUnwarpj*.

BigWarp

BigWarp è limitato dalla modalità di allineamento esclusivamente manuale, e dalla necessità di avere un computer recente non datato per poterlo sfruttare appieno. Il "Big" del suo nome è correlato al termine BigData, ed una delle sue funzionalità principali è proprio quella di lavorare con dataset di una certa grandezza.

ec-Clem

Il limite maggiore di *ec-CLEM* è l'enorme numero di finestre di dialogo utili ad utilizzare l'Autofinder, con le quali è facile perdere l'ordine degli eventi e ciò va a discapito dell'utente con medie competenze di analisi dati ed utilizzo di applicativi.

4.3.1 Elastix

"Elastix" ha sicuramente come limite quello di essere molto complesso da usare. La procedura d'installazione è prona ad errori, infatti serve scaricare sia il plugin dall'updater di *Fiji* che contiene solo la GUI, sia che scaricare il plugin vero e proprio da Github che contiene il vero programma, e specificarne il percorso della cartella una volta in fase d'esecuzione. La lista dei parametri è lunga e poco user-friendly.

5 Descrizione del modulo per allineamento automatico

Questo capitolo descrive il cuore di questo lavoro di Tesi, precisamente descrive il modulo implementato per feature l'allineamento automatico delle immagini. La funzionalità in questione è stata realizzata con l'ausilio di *"OpenCV"*, la più famosa libreria di *Computer Vision*. Tra gli argomenti trattati riporteremo ampie digressioni sulle scelte implementative fatte, utili alla costruzione della funzionalità.

5.1 Computer Vision

La *Computer Vision* è un sottoinsieme del campo dell'AI (*Artificial Intelligence*) ed è il tramite secondo il quale un'AI può prendere decisioni rispetto a ciò che "osserva". L'intenzione principale è imitare il nostro senso della vista, e come per noi esseri umani che sin da piccoli abbiamo imparato a classificare e distinguere le informazioni tramite retine, nervi ottici e corteccia visiva, la Computer Vision lo fa tramite algoritmi [30].

5.1.1 OpenCV Library

OpenCV (Open Source Computer Vision Library) è una libreria che contiene più di 2500 algoritmi di Computer Vision e *Machine Learning* in generale, la cui peculiarità è il fatto di essere Open Source. Il suo utilizzo è molto variegato e spazia dalla rilevazione di movimenti, oggetti e visi, al classificare i soggetti presenti in una sorgente video e ad operazioni di stitching tra immagini, etc. La libreria ha una community composta da più di 47 mila persone, ed è usata sia per motivi commerciali sia per ricerca scientifica. La maggior parte dei programmi legati alla Computer Vision basano la propria implementazione su questa libreria, e la possibilità di usarla in linguaggi come C++, Java, Python, Javascript e MATLAB e sistemi operativi come Windows, Linux, Mac OS X, iOS e Android, rendono la sua adozione la scelta preferita per applicazioni in cui è richiesto un uso intensivo di elaborazione di immagini e video in tempo reale [31].

5.1.2 OpenCV Extra Features Library

OpenCV contiene tantissimi moduli al suo interno e sono tutti usufruibili senza alcuna licenza anche per prodotti commerciali, ma ci sono delle eccezioni. Tutte le eccezioni sono contenute all'interno di una libreria chiamata *"OpenCV Contrib"* che contiene tutti i moduli con funzionalità che potrebbero essere non stabili, oppure la cui licenza è a pagamento per uso commerciale ma non per prodotti poi a sua volta resi open-source. In questa libreria si trovano due degli algoritmi in questo lavoro di Tesi al fine di creare il modulo per l'allineamento automatico delle immagini, *FREAK* e *"StarDetector"*.

5.1.3 Compilazione dalla sorgente

OpenCV è rilasciato in forma già compilata per Android, iOS e Windows mentre per qualsiasi altra piattaforma (o per quando è richiesto l'uso della libreria *OpenCV Contrib*) è necessario compilare personalmente il codice sorgente. Per implementare il modulo per l'allineamento automatico delle immagini, è stato necessario scaricare entrambi i sorgenti e per produrre un file JAR e le librerie relative alle piattaforme di esecuzione, ad esempio i file *.dll per Windows e i file *.dylib per i MacOS X con architettura ARM. A questo scopo, occorre avere una distribuzione Java installata sul proprio sistema operativo, scaricare il programma *"CMake"*, ed un terminale. Una volta scaricate le librerie *OpenCV* e *OpenCV Contrib* in formato zip ed estratte dagli archivi, è utile mantenere per comodità entrambe le cartelle risultanti all'interno di una cartella padre. Una volta completata questa operazione occorre aprire il terminale e specificare come percorso corrente la cartella padre. A questo punto si possono eseguire in sequenza i comandi Codice 8, Codice 9 e Codice 10. I file prodotti come risultato dell'esecuzione di questi comandi si troverà all'interno della cartella "Nome Cartella Padre/build/lib" e potranno essere usati nel proprio progetto. Nel nostro caso di studio, è bene ricordarsi di copiare questi files (e.g. opencv_core455.dll, opencv_java455.dll) all'interno della cartella "Fiji.app/libs".

```
mkdir build && cd build
```

Codice 8: Creazione cartella Build

```
cmake -DCMAKE_SYSTEM_PROCESSOR=arm64 \
-DCMAKE_OSX_ARCHITECTURES=arm64 \
-DWITH_OPENJPEG=OFF \
-DWITH_IPP=OFF \
-D CMAKE_BUILD_TYPE=Release \
-D CMAKE_INSTALL_PREFIX=/usr/local/opencv \
-D JAVA_INCLUDE_PATH=$JAVA_HOME/include \
-D JAVA_AWT_LIBRARY=$JAVA_HOME/jre/lib/amd64/libawt.so \
-D JAVA_JVM_LIBRARY=$JAVA_HOME/jre/lib/arm/server/libjvm.so \
-D OPENCV_EXTRA_MODULES_PATH=../opencv_contrib-4.5.5/modules \
-D WITH_FFMPEG=OFF \
-D WITH_OPENCL=OFF \
-D BUILD_opencv_java=ON \
-D OPENCV_ENABLE_NONFREE=ON \
-D BUILD_opencv_python2=OFF \
-D BUILD_opencv_python3=OFF \
-D BUILD_ZLIB=OFF \
-D BUILD_EXAMPLES=ON ../opencv-4.5.5
```

Codice 9: Esempio di creazione di build estrazione dalla sorgente per piattaforma MacOs X con processore M1

```
make -j8
```

Codice 10: Esempio compilazione sorgente per piattaforma MacOs X con processore M1, usufruibile anche per altre piattaforme UNIX e LINUX

5.2 Loader

DS4H Image Alignment è un plugin pensato per essere multipiattaforma e questo requisito deve rimanere tale anche per le librerie di terze parti importate. *OpenCV* per essere utilizzata richiede di essere caricata prima del plugin stesso, per ovviare a questo problema abbiamo usato uno static block all'interno della classe principale chiamata "ImageAlignment". Lo static block viene chiamato prima del costruttore della classe, e di conseguenza permette di gestire errori legati all'importazione prima di avviare il plugin. A questo punto si è scelto di usare lo **"Strategy Pattern"** che permette di scegliere in fase d'esecuzione la classe giusta, nel nostro caso ci ha permesso di definire due classi, una per le librerie per la piattaforma Windows e l'altra per la piattaforma Mac OS X basata su processori ARM.

```
static {
    ImageAlignment.loader();
}
```

Codice 11: Esempio di static block proveniente dal nostro caso di studio

5.3 AlignBuilder

La versione precedente del plugin *DS4H Image Alignment* permetteva solo la co-registrazione tramite corner points. Al fine di rispettare l'**Open-Closed Principle**, uno dei cinque principi **SOLID**, si è scelto di usare il **"Builder Pattern"**. I principi SOLID la base di un software nella programmazione ad oggetti. Il **Builder Pattern** richiede che ci sia un oggetto comune a tutte le classi che lo implementano/estendono, e nel caso di *DS4H Image Alignment* si è scelto di racchiudere il tutto in una abstract class chiamata *AbstractBuilder* che mantenesse le funzionalità comuni come il salvataggio delle immagini risultanti. Tale scelta permette di poter creare classi specifiche per l'allineamento, qualunque sia l'algoritmo scelto, senza dover modificare le classi già presenti.

5.4 Algoritmo di allineamento automatico

Nel nostro algoritmo si è scelto di fare la comparazione delle caratteristiche delle immagini a gruppi di due, dove la prima immagine è in realtà sempre la stessa e non viene modificata, mentre la seconda è quella a cui si applicano le deformazioni rispetto alle proporzioni della prima. Alla base della nuova modalità di allineamento automatico c'è stata la scelta di usare l'algoritmo già definito in precedenza chiamato *FREAK*. Tale algoritmo è stato sfruttato in fase di definizione delle corrispondenze delle caratteristiche tra due immagini. Per la fase antecedente ad essa si è scelto un algoritmo chiamato *"StarDetector"* che si occupa di ricercare le caratteristiche, algoritmo basato su [32], scelto poiché forniva risultanti soddisfacenti in tempi molto rapidi. Una volta costruiti i descrittori delle corrispondenze ed effettuata la fase di feature matching tramite distanza di Hamming, come già descritto nel Capitolo 1, si procede eliminando i falsi positivi filtrandoli attraverso l'esecuzione del "Lowe's ratio test", test dove si controlla che la distanza della prima corrispondenza non sia maggiore di una seconda corrispondenza moltiplicata per 0.75. Se si trovano almeno 4 corrispondenze si procedere con la deformazione. Il risultato ottenuto è visibile tramite la finestra di allineamento, le cui funzionalità sono già state descritte nel Capitolo 3.

5.5 Il codice in dettaglio

Partendo da un diagramma di flusso, descriveremo in dettaglio le principali parti di codice legate al modulo creato per l'allineamento automatico Figura 5.1.

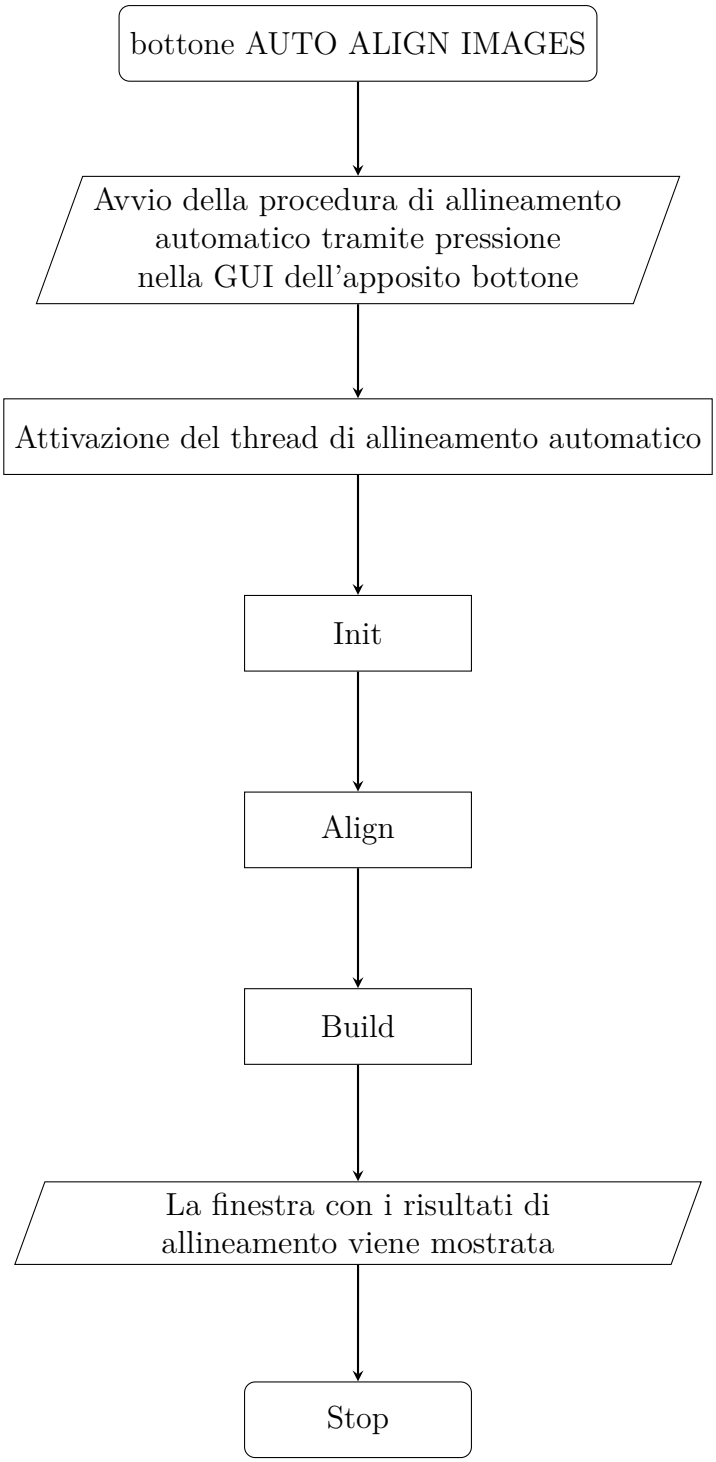

Figura 5.1: Diagramma di flusso modulo d'allineamento automatico

5.5.1 Evento allineamento automatico

Prima di tutto viene registrato l'evento legato alla pressione del bottone (Codice 12).

```
// MainDialog.java
final JButton btnAutoAlignment = new JButton("AUTO ALIGN IMAGES");
btnAutoAlignment.setToolTipText("Align the images automatically without thinking what it is needed to be
↪  done");
btnAutoAlignment.setEnabled(true);
btnAutoAlignment.addActionListener(e -> this.eventListener.onMainDialogEvent(new AutoAlignEvent()));
```

Codice 12: Porzione della classe MainDialog.java, classe che gestisce la GUI della finestra principale

Una volta premuto il bottone viene chiamato onMainDialogEvent, il quale ha al suo interno contiene un semplice switch case che in base al nome della classe richiama il metodo da eseguire. Nella porzione di codice presente in Codice 13 viene annotata la variabile di tipo *AbstractBuilder* inizializzata mediante la classe *FREAKBuilder*. *FREAKBuilder* è la classe che gestisce in toto l'allineamento automatico. Dopo aver mostrato una finestra di caricamento, utile per l'esperienza utente, viene chiamato il metodo *alignHandler* che gestisce l'ordine di chiamate ai metodi interni della classe *FREAKBuilder*. Prima di tutto si settano le immagini temporanee utili per le computazioni intermedie, poi viene chiamata *init* per l'inizializzazione, *align* per l'allineamento e *build* per l'importazione nella finestra di allineamento delle immagini co-registrate.

```
// ImageAlignment.java
private void autoAlign(AutoAlignEvent event) {
  AbstractBuilder builder = new FREAKBuilder(this.getLoadingDialog(), this.getManager(), event, this);
  builder.getLoadingDialog().showDialog();
  Utilities.setTimeout(() -> {
    try {
      this.alignHandler(builder);
    } catch (Exception e) {
      e.printStackTrace();
      IJ.showMessage(e.getMessage());
    }
    builder.getLoadingDialog().hideDialog();
  }, 10);
}

private void alignHandler(AbstractBuilder builder) {
  builder.setTempImages(this.tempImages);
  builder.init();
  builder.align();
  builder.build();
  this.alignDialog = builder.getAlignDialog();
  this.tempImages = builder.getTempImages();
}
```

Codice 13: Porzione della classe ImageAlignment.java, classe che gestisce l'evento proveniente da MainDialog.java

5.5.2 Init

Sono tre i task utili alla fase d'inizializzazione (Codice 14):

- L'importazione delle immagini, dove vengono presi i percorsi immagini presenti nello stack, ovvero quelle disponibili nella finestra principale. Vengono quindi ricaricate in funzione di OpenCV che utilizza il suo tipo nativo Mat, tipo di dato per gestire le immagini. Al contempo viene salvata una copia di "backup", che servirà poi nelle fasi successive.
- Si setta specifica la dimensione massima di un immagine, ciò definisce i limiti di altezza e larghezza per tutto lo stack di immagini.
- Inizializza la variabile virtualStack, al cui interno verranno aggiunte le immagini alla fine del processo di allineamento.

```
// FREAKBuilder.java
@Override
public void init() {
  this.importImages();
  this.setMaximumSize(new Dimension(this.getSourceImage().width(), this.getSourceImage().height()));
  this.setVirtualStack();
}

private void importImages() {
  for (ImageFile imageFile : this.getManager().getImageFiles()) {
    try {
      Mat matImage = imread(imageFile.getPathFile(), IMREAD_GRAYSCALE);
      this.pathMap.put(matImage.dataAddr(), imageFile.getPathFile());
      this.images.add(matImage);
    } catch (Exception e) {
      IJ.showMessage(e.getMessage());
    }
  }
}

// AbstractBuilder.java
private VirtualStack virtualStack;
private Dimension maximumSize;

protected void setMaximumSize(Dimension maximumSize) {
  this.maximumSize = maximumSize;
}

protected void setVirtualStack() {
  this.virtualStack = new VirtualStack(this.getMaximumSize().width, this.getMaximumSize().height,
↪  ColorModel.getRGBdefault(), IJ.getDir(TEMP_PATH));
}
```

Codice 14: Porzione delle classi FREAKBuilder.java e AbstractBuilder.java, per la gestione dell'inizializzazione

5.5.3 Align

Nella fase di allineamento Codice 15 si parte aggiungendo l'immagine "sorgente" al virtualStack, tramite il metodo *newProcessHandler*, che trasforma l'oggetto Mat in un oggetto ImagePlus per poter sfruttare i metodi di *ImageJ* (Codice 16). La comparazione viene fatta tra due immagini, e si ottengono da subito sia la matrice dei punti d'interesse sia il descrittore binario (Codice 14). Una volta eseguita il match tra i descrittori tramite

distanza di Hamming, viene eseguito il filtraggio dei falsi positivi Codice 19. Se i match rimanenti sono almeno quattro si continua a procedere nell'allineamento, dapprima si definiscono i limiti delle dimensioni delle immagini risultanti Codice 18, e per ottenere i coefficienti di trasformazione tra la seconda e la prima immagine si calcolano i valori della matrice di omografia (Codice 17). Terminata l'operazione si aggiunge l'immagine deformata allo stack virtuale.

```
@Override
public void align() {
    try {
        this.newProcessHandler(this.getSourceImage());
        for (int index = 1; index < images.size(); index++) {
        final Mat firstImage = this.getSourceImage();
        final Mat secondImage = images.get(index);
        final MatOfKeyPoint firstKeyPoints = this.getKeypoint(firstImage);
        final MatOfKeyPoint secondKeyPoints = this.getKeypoint(secondImage);
        final Mat firstDescriptor = this.getDescriptor(firstImage, firstKeyPoints);
        final Mat secondDescriptor = this.getDescriptor(secondImage, secondKeyPoints);
        // match
        final DescriptorMatcher matcher =
↪   DescriptorMatcher.create(DescriptorMatcher.BRUTEFORCE_HAMMING);
        // convert to float to use flann ( FREAK is a binary descriptor )
        if (firstDescriptor.type() != CV_8U) {
            firstDescriptor.convertTo(firstDescriptor, CV_8U);
        }
        if (secondDescriptor.type() != CV_8U) {
            secondDescriptor.convertTo(secondDescriptor, CV_8U);
        }
        // match descriptors and filter to avoid false positives
        List<MatOfDMatch> knnMatches = new ArrayList<>();
        try {
            matcher.knnMatch(firstDescriptor, secondDescriptor, knnMatches, 2);
        } catch (Exception e) {
            IJ.showMessage("Check all your images, one of them seems to have not valuable matches for
↪   our algorithm");
        }
        List<DMatch> goodMatches = getGoodMatches(knnMatches);
        if (goodMatches.size() > 4) {
            final MatOfPoint2f points = new MatOfPoint2f(this.getPointsArray(firstImage));
            Mat dest = new Mat();
            Core.perspectiveTransform(points, dest, this.getHomography(goodMatches,
↪   firstKeyPoints.toList(), secondKeyPoints.toList()));
            final Mat perspectiveM = Imgproc.getPerspectiveTransform(points, dest);
            Mat warpedImage = new Mat();
            // Literally takes the secondImage, the perspective transformation matrix, the size of the
↪   first image, then warps the second image to fit the first,
            Imgproc.warpPerspective(secondImage, warpedImage, perspectiveM, new Size(firstImage.width(),
↪   firstImage.height()), Imgproc.WARP_INVERSE_MAP, Core.BORDER_CONSTANT);
            // the image's data address changes after but the image it's the same, I've done it to
↪   retrieve the same path
            this.replaceKey(secondImage.dataAddr(), warpedImage.dataAddr());
            // then to all the things need to create a virtual stack of images
            this.newProcessHandler(warpedImage);
        } else {
            IJ.showMessage("Not enough matches");
        }
        }
    } catch (Exception e) {
        IJ.showMessage("Not all the images will be put in the aligned stack, something went wrong, check
↪   your image because it seems that we couldn't find a relation");
    }
}
```

Codice 15: Porzione delle classe FREAKBuilder.java, per la gestione dell'allineamento

```
private void newProcessHandler(Mat image) {
    ImageProcessor newProcessor = new ColorProcessor(image.width(), image.height());
    ImagePlus transformedImage = matToImagePlus(image);
    newProcessor.insert(transformedImage.getProcessor(), 0, 0);
    this.addToVirtualStack(new ImagePlus("", newProcessor), this.getVirtualStack());
}

private ImagePlus matToImagePlus(Mat image) {
    final int type = image.type();
    ImageProcessor result = null;
    if (type == CV_8UC1) {
        result = makeByteProcessor(image);
    } else {
        IJ.showMessage("Not supported for now");
    }
    return new ImagePlus("", result);
}

private ImageProcessor makeByteProcessor(Mat find) {
    final int w = find.cols();
    final int h = find.rows();
    ByteProcessor bp = new ByteProcessor(w, h);
    find.get(0, 0, (byte[]) bp.getPixels());
    return bp;
}

private Mat getSourceImage() {
    return this.images.get(0);
}
```

Codice 16: Porzione delle classe FREAKBuilder.java, per la conversione da Mat a ImagePlus

```
private Mat getHomography(List<DMatch> goodMatches, List<KeyPoint> firstKeyPoints, List<KeyPoint>
↪ secondKeyPoints) {
    final List<Point> obj = new ArrayList<>();
    final List<Point> scene = new ArrayList<>();
    final List<KeyPoint> listOfKeyPointsObject = new ArrayList<>(firstKeyPoints);
    final List<KeyPoint> listOfKeyPointsScene = new ArrayList<>(secondKeyPoints);
    for (int i = 0; i < goodMatches.size(); i++) {
        obj.add(listOfKeyPointsObject.get(goodMatches.get(i).queryIdx).pt);
        scene.add(listOfKeyPointsScene.get(goodMatches.get(i).trainIdx).pt);
    }
    MatOfPoint2f objMat = new MatOfPoint2f();
    MatOfPoint2f sceneMat = new MatOfPoint2f();
    objMat.fromList(obj);
    sceneMat.fromList(scene);
    final double freakThreshold = 0.005;
    return Calib3d.findHomography(objMat, sceneMat, Calib3d.RANSAC, freakThreshold);
}
```

Codice 17: Porzione delle classe FREAKBuilder.java, per l'ottenimento dell'omografia

```
private Point[] getPointsArray(Mat firstImage) {
    final Point[] pointsArray = new Point[4];
    final int width = firstImage.width();
    final int height = firstImage.height();
    pointsArray[0] = new Point(0, 0);
    pointsArray[1] = new Point(0, height);
    pointsArray[2] = new Point(width, height);
    pointsArray[3] = new Point(width, 0);
    return pointsArray;
}
```

Codice 18: Porzione delle classe FREAKBuilder.java, che gestisce i limiti delle dimensioni delle immagini in funzione di OpencCV

```
private List<DMatch> getGoodMatches(List<MatOfDMatch> knnMatches) {
    List<DMatch> listOfGoodMatches = new ArrayList<>();
    //- Filter matches using the Lowe's ratio test
    float ratioThresh = 0.75f;
    for (int i = 0; i < knnMatches.size(); i++) {
        if (knnMatches.get(i).rows() > 1) {
            DMatch[] matches = knnMatches.get(i).toArray();
            if (matches[0].distance < ratioThresh * matches[1].distance) {
                listOfGoodMatches.add(matches[0]);
            }
        }
    }
    return listOfGoodMatches;
}
```

Codice 19: Porzione delle classe FREAKBuilder.java, che esegue il filtraggio dei falsi positivi

```
private Mat getDescriptor(Mat image, MatOfKeyPoint keyPoints) {
    final Mat tempDescriptor = new Mat();
    final FREAK extractor = FREAK.create();
    extractor.compute(image, keyPoints, tempDescriptor);
    return tempDescriptor;
}

private MatOfKeyPoint getKeypoint(Mat image) {
    final MatOfKeyPoint tempKeypoint = new MatOfKeyPoint();
    final StarDetector detector = StarDetector.create();
    detector.detect(image, tempKeypoint);
    return tempKeypoint;
}
```

Codice 20: Porzione delle classe FREAKBuilder.java, nella quale si ottiene la matrice dei punti d'interesse e il descrittore binario

5.5.4 Build

La fase finale riguarda esclusivamente la finestra di allineamento ed il salvataggio automatico dei file TIFF risultanti dalla procedura di allineamento Codice 21

```
@Override
public void build() {
    try {
        this.setTransformedImagesStack(new ImagePlus("", this.getVirtualStack()));
        String filePath = IJ.getDir(TEMP_PATH) + this.getTransformedImagesStack().hashCode() + TIFF_EXT;
        new ImageConverter(this.getTransformedImagesStack()).convertToRGB();
        new FileSaver(this.getTransformedImagesStack()).saveAsTiff(filePath);
        this.getTempImages().add(filePath);
        this.getLoadingDialog().hideDialog();
        this.setAlignDialog(new AlignDialog(this.getTransformedImagesStack(), this.getListener()));
        this.getAlignDialog().pack();
        this.getAlignDialog().setVisible(true);
    } catch (Exception e) {
        IJ.showMessage(e.getMessage());
    }
    this.getLoadingDialog().hideDialog();
}
```

Codice 21: Porzione delle classe FREAKBuilder.java, per la gestione della visualizzazione a finestra

5.6 Reload Stack

Il nostro algoritmo di allineamento automatico riesce a registrare le immagini con ottimi risultati fino a quando la variazione d'intensità tra le immagini stesse non risulta davvero eccessiva. L'unico caso pratico in cui abbiamo riscontrato dei problemi è stato legato all'utilizzo di immagini *DIC*. Per ovviare a questa problema si possono effettuare questi passi:

1. Usare l'allineamento automatico per tutte le immagini non DIC.
2. Usare la funzione "Reuse as source" che riusa tali immagini nello stack principale.
3. Usare infine l'allineamento via corner points.
4. Consiglio: il plugin è perfettamente integrato con *Fiji* e quindi si possono sempre sfruttare tutte le funzionalità di *Fiji* legate alla modifica delle immagini, come ad esempio cambiarne il contrasto della immagine DIC. È importante notare che l'allineamento finale avverrà comunque tra le immagini originali non modificate.

6 Conclusioni

In questo progetto di Tesi abbiamo affrontato l'argomento "registrazione di immagini multimodali". In particolare, ci siamo soffermati all'analisi della registrazione di immagini in due dimensioni, senza necessità di attuare correzioni elastiche per correggere eventuali deformazioni delle stesse. Il problema è un problema noto in microscopia perché spesso per analizzare un provino istologico vengono utilizzati differenti marcatori tumorali che richiedono differenti preparativi. Si acquisiscono quindi sequenze di immagini utilizzando differenti coloranti e per poter avere una visione globale e proseguire con analisi di co-localizzazione occorre poi unire le immagini acquisite in un unico riferimento.

Con lo scopo di rendere il processo di allineamento delle sequenze di immagini acquisite estremamente semplice, abbiamo sviluppato *DS4H Image Alignment*, un tool open-source estremamente user friendly, attualmente distribuito come plugin di *Fiji*, una piattaforma ben conosciuta da medici e biologi. In particolare, partendo da una versione iniziale sviluppata in precedenza dal gruppo di ricerca *DS4H*, abbiamo esteso le funzionalità del tool integrando un modulo per la registrazione automatica delle immagini. Infatti, in precedenza il tool presentava una unica possibilità di registrazione che prevedeva la definizione manuale di corner point corrispondenti tra immagini seguenti e un successivo allenamento basata sulla stima ai minimi quadrati dei parametri di registrazione di una matrice affine. Oltre allo sviluppo del modulo per l'allineamento automatico, l'interfaccia grafica è stata rivoluzionata al fine di integrare una serie di opportunità per una più semplice gestione dei corner points e delle eventuali correzioni di registrazione.

Attualmente il tool è disponibile per due piattaforme, Windows e Mac Os X (solo per Apple Silicon), scaricabili dalla sezione Release della repository pubblica Github: UniBoDS4H/DS4H-Image-Alignment.

Il primo degli obiettivi futuri di questo progetto di Tesi sarà quello di rendere disponibile il tool anche per le piattaforme Linux e Mac Os X (con processori Intel). Inoltre, integreremo il plugin anche in *ImageJ*, attuando un refactoring completo del codice. Al momento non è presente una divisione dei compiti in maniera netta, e il codice risulta strettamente accoppiato. Ciò significa che qualsiasi tentativo di estensione risulta oneroso in termini di tempo. Serve quindi dare una struttura chiara su cui poter lavorare per estendere con

semplicità le funzionalità attualmente presenti. In particolare, l'integrazione di *ImageJ2* sarà fatta attraverso l'utilizzo dei Services, con i quali si può destrutturare l'architettura presente e creare moduli, classi e metodi con una sola responsabilità. *ImageJ2*, attraverso la libreria *ImageJ Ops*, permette di rendere disponibile il plugin non solo su *ImageJ*, introducendo così una quasi perfetta retrocompatibilità, ma anche su software come *CellProfiler* e *Omero*, rendendo così il software altamente interoperabile. Infine, grazie alla presenza in *ImageJ/Fiji* del plugin *BigWarp*, integreremo la possibilità di gestire deformazioni elastiche, eventualmente considerando le performanti *"Thin plate spline"* piuttosto che le *BSpline*, così da fornire un'alternativa ottimizzata rispetto alle soluzioni ora presenti in letteratura.

Concludendo, nonostante *DS4H Image Alignment* sia da considerare ancora come un tool in fase di sviluppo, attualmente risulta essere la soluzione più semplice per biologi e medici che hanno necessita di registrare immagini 2D ottenute con seguenti acquisizioni al microscopio. Il tool è già stato utilizzato e citato in un primo articolo scientifico pubblicato in Frontiers in Immunology ed una descrizione dettagliata dello stesso verrà a breve sottomessa ad una nuova rivista. Nel frattempo, e gratuitamente disponibile per la comunità ed è quotidianamente utilizzato presso l'Istituto Romagnolo per lo Studio dei Tumori "Dino Amadori" IRCCS di Meldola.

List of Listings

Elenco delle figure

Bibliografia

[1] S. Belli, «Studio e realizzazione di un plugin per l'allineamento di immagini microscopiche», tesi di dott., UniBo, 2019. indirizzo: http://amslaurea.unibo.it/19123/.

[2] J. Bulgarelli, M. Tazzari, A. M. Granato, L. Ridolfi, S. Maiocchi, F. de Rosa, M. Petrini, E. Pancisi, G. Gentili, B. Vergani, F. Piccinini, A. Carbonaro, B. E. Leone, G. Foschi, V. Ancarani, M. Framarini e M. Guidoboni, «Dendritic cell vaccination in metastatic melanoma turns "non-T cell inflamed" into "T-cell inflamed" tumors», en, *Front. Immunol.*, vol. 10, p. 2353, ott. 2019.

[3] E. Treccani. «istologia». (2022), indirizzo: https://www.treccani.it/enciclopedia/istologia/.

[4] H. A. Alturkistani, F. M. Tashkandi e Z. M. Mohammedsaleh, «Histological stains: A literature review and case study», en, *Glob. J. Health Sci.*, vol. 8, n. 3, pp. 72–79, giu. 2015.

[5] J. Borovec, J. Kybic, I. Arganda-Carreras, D. V. Sorokin, G. Bueno, A. V. Khvostikov, S. Bakas, E. I.-C. Chang, S. Heldmann, K. Kartasalo, L. Latonen, J. Lotz, M. Noga, S. Pati, K. Punithakumar, P. Ruusuvuori, A. Skalski, N. Tahmasebi, M. Valkonen, L. Venet, Y. Wang, N. Weiss, M. Wodzinski, Y. Xiang, Y. Xu, Y. Yan, P. Yushkevich, S. Zhao e A. Muñoz-Barrutia, «ANHIR: Automatic Non-Rigid Histological Image Registration Challenge», *IEEE Transactions on Medical Imaging*, vol. 39, n. 10, pp. 3042–3052, 2020, ISSN: 1558-254X. DOI: 10.1109/TMI.2020.2986331.

[6] D. G. Lowe, «Distinctive Image Features from Scale-Invariant Keypoints», *International Journal of Computer Vision*, vol. 60, n. 2, pp. 91–110, nov. 2004, ISSN: 1573-1405. DOI: 10.1023/B:VISI.0000029664.99615.94. indirizzo: https://doi.org/10.1023/B:VISI.0000029664.99615.94.

[7] H. Bay, A. Ess, T. Tuytelaars e L. Van Gool, «Speeded-Up Robust Features (SURF)», *Computer Vision and Image Understanding*, vol. 110, n. 3, pp. 346–359, 2008, Similarity Matching in Computer Vision and Multimedia, ISSN: 1077-3142. DOI: https://doi.org/10.1016/j.cviu.2007.09.014. indirizzo: https://www.sciencedirect.com/science/article/pii/S1077314207001555.

[8] S. Leutenegger, M. Chli e R. Y. Siegwart, «BRISK: Binary Robust invariant scalable keypoints», in *2011 International Conference on Computer Vision*, 2011, pp. 2548–2555. DOI: 10.1109/ICCV.2011.6126542.

[9] Y. Liu, H. Zhang, H. Guo e N. N. Xiong, «A FAST-BRISK feature detector with depth information», en, *Sensors (Basel)*, vol. 18, n. 11, p. 3908, nov. 2018.

[10] E. Mair, G. D. Hager, D. Burschka, M. Suppa e G. Hirzinger, «Adaptive and Generic Corner Detection Based on the Accelerated Segment Test», in *Computer Vision – ECCV 2010*, K. Daniilidis, P. Maragos e N. Paragios, cur., Berlin, Heidelberg: Springer Berlin Heidelberg, 2010, pp. 183–196, ISBN: 978-3-642-15552-9.

[11] E. Rosten, R. Porter e T. Drummond, «Faster and better: a machine learning approach to corner detection», en, *IEEE Trans. Pattern Anal. Mach. Intell.*, vol. 32, n. 1, pp. 105–119, gen. 2010.

[12] A. Alahi, R. Ortiz e P. Vandergheynst, «FREAK: Fast Retina Keypoint», in *2012 IEEE Conference on Computer Vision and Pattern Recognition*, 2012, pp. 510–517. DOI: 10.1109/CVPR.2012.6247715.

[13] C. A. Schneider, W. S. Rasband e K. W. Eliceiri, «NIH Image to ImageJ: 25 years of image analysis», *Nature Methods*, vol. 9, n. 7, pp. 671–675, lug. 2012, ISSN: 1548-7105. DOI: 10.1038/nmeth.2089. indirizzo: https://doi.org/10.1038/nmeth.2089.

[14] *Legacy CheatSheet : ImageJ 1.x to ImageJ2 transition*, https://github.com/mpicbg-scicomp/ij2course-images/blob/master/slides/ij_legacy_cheetsheet.pdf, Accessed: 2022-04-30.

[15] J. Schindelin, C. T. Rueden, M. C. Hiner e K. W. Eliceiri, «The ImageJ ecosystem: An open platform for biomedical image analysis», *Molecular Reproduction and Development*, vol. 82, n. 7-8, pp. 518–529, 2015. DOI: https://doi.org/10.1002/mrd.22489. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1002/mrd.22489. indirizzo: https://onlinelibrary.wiley.com/doi/abs/10.1002/mrd.22489.

[16] C. T. Rueden, J. Schindelin, M. C. Hiner, B. E. DeZonia, A. E. Walter, E. T. Arena e K. W. Eliceiri, «ImageJ2: ImageJ for the next generation of scientific image data», *BMC Bioinformatics*, vol. 18, n. 1, p. 529, nov. 2017, ISSN: 1471-2105. DOI: 10.1186/s12859-017-1934-z. indirizzo: https://doi.org/10.1186/s12859-017-1934-z.

[17] M. C. Hiner, C. T. Rueden e K. W. Eliceiri, «SCIFIO: an extensible framework to support scientific image formats», *BMC Bioinformatics*, vol. 17, n. 1, p. 521, dic. 2016, ISSN: 1471-2105. DOI: 10.1186/s12859-016-1383-0. indirizzo: https://doi.org/10.1186/s12859-016-1383-0.

[18] T. Pietzsch, S. Preibisch, P. Tomančák e S. Saalfeld, «ImgLib2—generic image processing in Java», *Bioinformatics*, vol. 28, n. 22, pp. 3009–3011, set. 2012, ISSN: 1367-4803. DOI: 10.1093/bioinformatics/bts543. eprint: https://academic.oup.com/bioinformatics/article-pdf/28/22/3009/16908600/bts543.pdf. indirizzo: https://doi.org/10.1093/bioinformatics/bts543.

[19] J. Schindelin, I. Arganda-Carreras, E. Frise, V. Kaynig, M. Longair, T. Pietzsch, S. Preibisch, C. Rueden, S. Saalfeld, B. Schmid, J.-Y. Tinevez, D. J. White, V. Hartenstein, K. Eliceiri, P. Tomancak e A. Cardona, «Fiji: an open-source platform for biological-image analysis», *Nature Methods*, vol. 9, n. 7, pp. 676–682, lug. 2012, ISSN: 1548-7105. DOI: 10.1038/nmeth.2019. indirizzo: https://doi.org/10.1038/nmeth.2019.

[20] M. Linkert, C. T. Rueden, C. Allan, J.-M. Burel, W. Moore, A. Patterson, B. Loranger, J. Moore, C. Neves, D. MacDonald, A. Tarkowska, C. Sticco, E. Hill, M. Rossner, K. W. Eliceiri e J. R. Swedlow, «Metadata matters: access to image data in the real world», *Journal of Cell Biology*, vol. 189, n. 5, pp. 777–782, mag. 2010, ISSN: 0021-9525. DOI: 10.1083/jcb.201004104. eprint: https://rupress.org/jcb/article-pdf/189/5/777/1346115/jcb_201004104.pdf. indirizzo: https://doi.org/10.1083/jcb.201004104.

[21] C. Allan, J.-M. Burel, J. Moore, C. Blackburn, M. Linkert, S. Loynton, D. MacDonald, W. J. Moore, C. Neves, A. Patterson, M. Porter, A. Tarkowska, B. Loranger, J. Avondo, I. Lagerstedt, L. Lianas, S. Leo, K. Hands, R. T. Hay, A. Patwardhan, C. Best, G. J. Kleywegt, G. Zanetti e J. R. Swedlow, «OMERO: flexible, model-driven data management for experimental biology», *Nature Methods*, vol. 9, n. 3, pp. 245–253, mar. 2012, ISSN: 1548-7105. DOI: 10.1038/nmeth.1896. indirizzo: https://doi.org/10.1038/nmeth.1896.

[22] S. Li, S. Besson, C. Blackburn, M. Carroll, R. K. Ferguson, H. Flynn, K. Gillen, R. Leigh, D. Lindner, M. Linkert, W. J. Moore, B. Ramalingam, E. Rozbicki, G. Rustici, A. Tarkowska, P. Walczysko, E. Williams, C. Allan, J.-M. Burel, J. Moore e J. R. Swedlow, «Metadata management for high content screening in OMERO», *Methods*, vol. 96, pp. 27–32, 2016, High-throughput Imaging, ISSN: 1046-2023. DOI: https://doi.org/10.1016/j.ymeth.2015.10.006. indirizzo: https://www.sciencedirect.com/science/article/pii/S1046202315301250.

[23] A. Cardona, S. Saalfeld, J. Schindelin, I. Arganda-Carreras, S. Preibisch, M. Longair, P. Tomancak, V. Hartenstein e R. J. Douglas, «TrakEM2 Software for Neural Circuit Reconstruction», *PLOS ONE*, vol. 7, n. 6, pp. 1–8, giu. 2012. DOI: 10.1371/journal.pone.0038011. indirizzo: https://doi.org/10.1371/journal.pone.0038011.

[24] S. Schaefer, T. McPhail e J. Warren, «Image Deformation Using Moving Least Squares», *ACM Trans. Graph.*, vol. 25, n. 3, pp. 533–540, lug. 2006, ISSN: 0730-0301. DOI: 10.1145/1141911.1141920. indirizzo: https://doi.org/10.1145/1141911.1141920.

[25] C. Joe, M. Michael, B. Tim, B. Kiran, K. Walid, B. Peter e B. Mary, «MIST: Accurate and Scalable Microscopy Image Stitching Tool with Stage Modeling and Error Minimization», *Scientific Reports*, vol. 7, n. 4098, 2017. DOI: 10.1038/s41598-017-04567-y. indirizzo: https://doi.org/10.1038/s41598-017-04567-y.

[26] F. ROHDE, U.-D. BRAUMANN e M. SCHMIDT, «Correlia: an ImageJ plug-in to co-register and visualise multimodal correlative micrographs», *Journal of Microscopy*, vol. 280, n. 1, pp. 3–11, 2020. DOI: https://doi.org/10.1111/jmi.12928. eprint: https://onlinelibrary.wiley.com/doi/pdf/10.1111/jmi.12928. indirizzo: https://onlinelibrary.wiley.com/doi/abs/10.1111/jmi.12928.

[27] J. A. Bogovic, P. Hanslovsky, A. Wong e S. Saalfeld, «Robust registration of calcium images by learned contrast synthesis», in *2016 IEEE 13th International Symposium on Biomedical Imaging (ISBI)*, 2016, pp. 1123–1126. DOI: 10.1109/ISBI.2016.7493463.

[28] P. Paul-Gilloteaux, X. Heiligenstein, M. Belle, M.-C. Domart, B. Larijani, L. Collinson, G. Raposo e J. Salamero, «eC-CLEM: flexible multidimensional registration software for correlative microscopies», *Nature Methods*, vol. 14, n. 2, pp. 102–103, feb. 2017, ISSN: 1548-7105. DOI: 10.1038/nmeth.4170. indirizzo: https://doi.org/10.1038/nmeth.4170.

[29] S. Klein, M. Staring, K. Murphy, M. A. Viergever e J. P. W. Pluim, «<emphasis emphasistype="mono">elastix</emphasis>: A Toolbox for Intensity-Based Medical Image Registration», *IEEE Transactions on Medical Imaging*, vol. 29, n. 1, pp. 196–205, 2010. DOI: 10.1109/TMI.2009.2035616.

[30] *Cos'è la Computer Vision?*, https://www.ibm.com/it-it/topics/computer-vision, Accessed: 2022-05-01.

[31] G. Bradski, «The OpenCV Library», *Dr. Dobb's Journal of Software Tools*, 2000.

[32] M. Agrawal, K. Konolige e M. R. Blas, «CenSurE: Center Surround Extremas for Realtime Feature Detection and Matching», in *Computer Vision – ECCV 2008*, D. Forsyth, P. Torr e A. Zisserman, cur., Berlin, Heidelberg: Springer Berlin Heidelberg, 2008, pp. 102–115, ISBN: 978-3-540-88693-8.

Ringraziamenti

Per prima cosa vorrei ringraziare la Professoressa Antonella Carbonaro, la mia Relatrice, che mi ha dato la possibilità di mostrare le mie competenze dandomi la responsabilità di partecipare a questo bellissimo progetto. Insieme a Lei, vorrei ringraziare uno dei miei Correlatori, il Professore Filippo Piccinini, senza le sue direttive e l'aiuto costante probabilmente non ce l'avrei fatta. Davvero grazie di tutto, è stata lunga ma siamo riusciti a portare a casa un ottimo risultato. Ringrazio anche i Professori Giovanni Martinelli e Gastone Castellani per aver reso possibile il progetto, consentendo l'utilizzo di immagini reali acquisite presso l'Istituto Romagnolo per lo Studio dei Tumori "Dino Amadori" IRCCS di Meldola.

Vorrei ringraziare inoltre i Professori con cui ho dato i miei due ultimi esami, ovvero il Professore Mirko Viroli e il Professore Alessandro Ricci che hanno riconosciuto i miei sforzi. Ringrazio i miei amici Valentina Gabellini, Giuliano Esposito, Rocco Mantovani, Giulia Belleffi, Silvia Bezzecchi, e forse è meglio che mi limiti a scrivere nomi e cognomi perché la lista finirebbe nel 2032, che mi hanno sopportato, supportato, guidato e insegnato in questo lunghissimo percorso.

Ringrazio chi negli anni mi ha permesso di lavorare e accumulare esperienza come programmatore, soprattutto i miei ex-colleghi di Corsi.it, che mi hanno sostenuto e aiutato ad uscire dalla depressione, mi hanno aiutato a gestire l'ansia e mi hanno reso una persona sicuramente migliore.

Last but not least, la mia famiglia, il mio nucleo, mia sorella Uarda, mio padre Eduart e mia madre Mimoza, che nonostante tutte le avversità, e in generale nonostante **TUTTO**, che detta così fa molto film strappa lacrime, ma è vero, mi ha dato tutto l'aiuto di cui avevo bisogno, mi ha insegnato a reagire ai problemi e non lasciare che gli stessi ti sommergano senza più via d'uscita. Vorrei ringraziare in ultimo la mia Professoressa di Matematica del liceo, che alla fine del terzo anno mi bocciò e quando arrivai al quinto anno, nonostante non fosse più la mia docente, mi fermò e mi consiglio questo percorso universitario, vedendo in me capacità che non credevo di possedere.