**Alma Mater Studiorum – Università di Bologna – Campus di Cesena**

Dipartimento Informatica – Scienza e Ingegneria (DISI)

Corso di Laurea in Ingegneria e Scienze Informatiche

# DS4H Image Alignment tool: algoritmi per allineamento automatico con estensione dei tool competitor e possibilità di correzioni elastiche includendo gestione ottimizzata di immagini multicanale

Tesi in: Programmazione

**Relatore**

Prof.ssa Antonella Carbonaro

**Presentata da**

Matteo Iorio

**Correlatori**

Prof. Filippo Piccinini

Prof. Martinelli Giovanni

Prof. Gastone Castellani

**Anno Accademico 2022 / 2023**

“*A cosa posso paragonare*

*il mondo e la vita dell’uomo?*

*Al riflesso della luna*

*nella goccia di rugiada, scossa*

*dal becco di una gru.”*

*Dōgen*

# Parole Chiave

Istologia

Microscopia

Allineamento multimodale

Registrazione automatica

Open-source tool

# Abstract (English version)

Most of the cancer studies begin by analyzing histological specimens or tissue samples obtained from patients. Specific markers or selective dyes are applied to the section of the sample to be studied, allowing for the examination of specific parts and subcellular compartments for gathering more information from the sample. However, these markers cannot always be applied simultaneously and are typically used sequentially after washing the sample. The resulting images must then be aligned in order to conduct colocalization studies and simulate the simultaneous acquisition of different signals. Unfortunately, there is no standardized procedure for aligning these images, and manual alignment is time-consuming and prone to errors. The use of a software could streamline the process and increase the reliability of the final analysis. In particular, *DS4H Image Alignment* is an open-source, user-friendly plug-in implemented for *ImageJ*/*Fiji* and designed to align grayscale multimodal images with minimal user interaction.

# Abstract (versione italiana)

Molti degli studi sul cancro iniziano con l'analisi di campioni istologici, cioè campioni di tessuto prelevati dal paziente. Attraverso l'uso di marcatori specifici, ovvero coloranti selettivi applicati alla sezione da analizzare, vengono studiate parti specifiche del campione ed ottenute più informazioni dal campione stesso. Tuttavia, questi marcatori non possono sempre essere applicati contemporaneamente e spesso vengono utilizzati in sequenza dopo un lavaggio del campione. Per poter condurre studi di colocalizzazione, le immagini ottenute devono quindi essere allineate simulando l'acquisizione parallela dei diversi segnali. Purtroppo, non esiste una procedura standard per l'allineamento di queste immagini e l'allineamento manuale richiede tempo ed è soggetto a possibili errori. L'utilizzo di un software potrebbe rendere il processo più rapido e affidabile. In particolare, *DS4H Image Alignment* è un plug-in open-source, user-friendly implementato per *ImageJ*/*Fiji* che permette di allineare immagini multimodali con una richiesta di minima iterazione da parte dell'utente.

# Indice

# Introduzione

Questo progetto di Tesi è stato sviluppato all'interno del gruppo di ricerca "*Data Science for Health*" (DS4H), composto da ricercatori e professori dell'Università di Bologna (UniBo) e dell'*Istituto Romagnolo dei Tumori Dino Amadori* (IRST) di Meldola (FC). Il gruppo DS4H si propone di coordinare professionisti e risorse provenienti dai settori dell'informatica, dell'ingegneria dell'informazione, della fisica, nonché dalle discipline biomediche, della biologia, della chimica e della medicina. In particolare, il progetto di tesi intitolato "*DS4H Image Alignment tool: algoritmi per allineamento automatico con estensione dei tool competitor e possibilità di correzioni elastiche includendo gestione ottimizzata di immagini multicanale*" si basa sui lavori pubblicati in una serie di Tesi precedenti che hanno portato alla creazione delle prime versioni del software chiamato "*DS4H Image Alignment*". L'obiettivo di questo progetto è consentire la registrazione di immagini multimodali 2D acquisite con microscopi a campo largo, al fine di condurre successivamente studi di colocalizzazione dei segnali intracellulari, migliorando così la valutazione dei campioni istologici in esame.

In particolare, *DS4H Image Alignment* risponde alle richieste originariamente formulate da medici e biologi dell'IRST. Per comprendere il contesto operativo, è importante spiegare come i professionisti dell'IRST valutano tipicamente i tumori dei pazienti quando dispongono di una biopsia del tessuto. Di solito, si inizia creando provini istologici dai campioni di tessuto prelevati dal paziente, a cui vengono applicati marcatori specifici per identificare le cellule con caratteristiche tumorali particolari. Questi marcatori sono spesso sonde fluorescenti visibili attraverso un microscopio a fluorescenza. Inoltre, per ottenere maggiori informazioni dal campione, vengono frequentemente utilizzati più marcatori. Tuttavia, questi marcatori non possono sempre essere applicati contemporaneamente e vengono spesso utilizzati in sequenza dopo un lavaggio del campione. Dopo l'applicazione dei coloranti, vengono solitamente ottenute immagini fluorescenti utilizzando microscopi a campo largo. Tuttavia, queste immagini devono essere allineate per condurre studi di colocalizzazione, simulando l'acquisizione parallela dei vari segnali (come d'esempio il nucleo, il citoplasma e la membrana). Purtroppo, non esiste una procedura standard per l'allineamento di queste immagini. L'allineamento manuale richiede molto tempo ed è soggetto a possibili errori. Un software potrebbe rendere il processo più rapido e affidabile, consentendo il riconoscimento delle aree comuni e l'allineamento dei segnali. Da questa necessità è nato *DS4H Image Alignment*, con l'obiettivo di colmare questa lacuna tecnica e automatizzare il processo nell'ambito dell'istologia. L'obiettivo finale è quello di ottenere un'analisi rapida e precisa dei

campioni istologici attraverso uno strumento di allineamento delle immagini acquisite con diverse tecniche e colorazioni (multi-modalità).

Il riconoscimento del problema e le prime ipotesi di soluzione sono state formulate dalla Prof.ssa Antonella Carbonaro e dal Prof. Filippo Piccinini, successivamente sviluppate utilizzando immagini ottenute da campioni istologici di pazienti reali. Queste immagini sono state fornite dal Prof. *Giovanni Martinelli*, direttore scientifico dell'IRST di Meldola, e dal Prof. *Gastone Castellani*, direttore della Scuola di Specializzazione in Fisica Medica dell'Università di Bologna.

Questo progetto di tesi è stato svolto in collaborazione con un altro tesista del Corso di Laurea in Ingegneria e Scienze Informatiche, Vincenzi Fabio. A livello pratico, in questo abbiamo quotidianamente collaborato in tutti i seguenti tasks:

A) Implementazione seguendo il pattern *MVC*;
B) Studio dei tool dello stato dell'arte;
C) Implementazione e gestione dei *corner point*;
D) Implementazione algoritmi Traslativo, Proiettivo ed Affine per allineamento manuale;
E) Implementazione algoritmi SIFT e SURF per allineamenti automatico;
F) Integrazione del plugin *bUnwarpJ* per correzione deformazioni elastiche;
G) Gestione ottimizzata di immagini *big data*;
H) Gestione ottimizzata salvataggio progetto;
I) Gestione ottimizzata riutilizzo delle immagini per allineamenti successivi;
J) Validazione del tool usando dataset di test;
K) Release codice sorgente e *standalone* per Windows, Mac e Linux.

Tuttavia, io sono responsabile e mi sono occupato in maniera più specifica dei punti B, E, F, G, H, K. sviluppando tre moduli principali:

(I) Il primo modulo consente un allineamento automatico ottimizzato delle immagini senza perdita di informazioni. Attraverso degli algoritmi automatici vengono individuati una serie di *keypoints.* Una volta che i *keypoints* sono stati individuati, le immagini vengono sottoposte a una fase di pre-elaborazione, in cui la dimensione dell'immagine target viene aumentata *aggiungendo un preciso numero di pixel di margine*. Successivamente, tutte le immagini vengono allineate utilizzando l'algoritmo selezionato. Questo approccio evita la perdita di informazioni preziose sulle immagini oggetto di studio.

(II) Il secondo modulo affronta il problema derivante dal costante lavaggio dei campioni dopo l'applicazione di un colorante. In particolare, ad ogni risciacquo, il tessuto in studio tende a formare piccole pieghe e deformazioni. Per risolvere questo problema, è necessario utilizzare un algoritmo di deformazione elastica. Nel nostro caso è stato

utilizzato la libreria conosciuta con il nome di "*bUnwarpJ*". Grazie a questo algoritmo, siamo stati in grado di mitigare tali deformazioni elastiche e ad allineare le immagini contrastando le deformazioni causate dai lavaggi.

(III) Il terzo modulo espande le funzionalità del software per consentire la gestione di immagini a colori senza la necessità di convertirle in toni di grigio. Il nostro applicativo è in grado di accettare qualsiasi tipo di immagine in input con qualsiasi tipo di *bioFormat* e livello di toni (e.g. 8-bit, 16-bit, 32-bit).

Nei prossimi capitoli di questo documento vengono descritte le soluzioni specifiche e dettagliate proposte per lo sviluppo dei moduli sopra menzionati.

# 1 Provini istologici: importanza e preparazione

L'istologia è un settore della biologia che si occupa dell'analisi della struttura microscopica e ultramicroscopica dei tessuti animali e vegetali. Un campione di tessuto utilizzato per queste analisi è chiamato provino istologico. In questo documento, con il termine "*provino istologico*" ci riferiamo a campioni prelevati da pazienti con lo scopo di monitorare e studiare patologie, nonché identificare malattie per determinare il trattamento più appropriato.

Nel corso degli anni, le tecniche di studio dei campioni istologici sono state perfezionate, con l'adozione di metodologie che consentono una migliore e più duratura conservazione dei campioni stessi, oltre a una resa informativa ottimale per le analisi. In particolare, un provino istologico deve essere preparato seguendo cinque fasi fondamentali successive, chiamate:

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

Durante il processo di *fissazione*, si creano condizioni ottimali per il provino istologico, simili a quelle fisiologiche, al fine di conservare il tessuto e rafforzare la struttura cellulare attraverso una cross-correlazione irreversibile delle proteine. La fissazione può essere eseguita immergendo il campione in una sostanza biologica come la formalina o in un fluido per il congelamento. Successivamente, il campione sarà estratto da tale materiale per il sezionamento.

Successivamente, si procede con la disidratazione, un processo finalizzato alla rimozione dell'acqua dal campione sostituendola tipicamente con molecole di etanolo. Una volta completata la disidratazione, si utilizza lo xilene per rimuovere l'etanolo dal tessuto, in modo che possa essere preparato per il taglio al microtomo. Questa fase prevede l'uso di alcoli con una crescente intensità e volume per evitare danni alle cellule.

In seguito, si procede con *l'inclusione*, che è una fase in cui il tessuto viene solidificato utilizzando sostanze chimiche specifiche, di solito paraffina. Questo processo implica l'esposizione del campione al calore seguito dal congelamento per prevenire il deterioramento. Poiché la paraffina e

l'etanolo non sono miscibili tra loro, viene utilizzato un fluido che può mescolarsi con entrambi per "pulire" il campione prima dell'inclusione.

Nella fase successiva, chiamata *sezionamento*, il tessuto viene tagliato utilizzando un microtomo o un criostato, che sono strumenti di precisione appositamente progettati per questa operazione. L'obiettivo è ottenere sezioni sottili sufficienti, di solito intorno ai cinque micrometri, affinché la luce del microscopio possa attraversarle agevolmente per consentirne l'analisi.

Infine, si procede con *l'applicazione di diverse coloranti* al fine di evidenziare parti specifiche e proprietà del tessuto, aumentandone il contrasto e consentendo di distinguere tali parti dagli altri componenti del materiale biologico. Prima dell'applicazione dei coloranti, il campione viene sottoposto a un processo di deparaffinazione e reidratazione, in modo che possa assorbire le tinture. Successivamente, il campione viene risciacquato e disidratato una volta completata la procedura. Questa fase è nota come *colorazione* e può alterare la sezione del campione, richiedendo quindi un'ulteriore fase di lavaggio o la sua sostituzione prima di ripetere la procedura per lo studio di diverse parti utilizzando coloranti diversi [1][2].

Le seguenti illustrazioni mostrano i risultati ottenuti acquisendo immagini del medesimo campione utilizzando un microscopio a campo largo. Il tessuto del campione proviene dal rene di un modello murino ed è stato sottoposto a tutte le fasi descritte in precedenza. In particolare, sono state acquisite immagini utilizzando la tecnica del Differential Interference Contrast (DIC) e immagini fluorescenti dopo l'applicazione di tre diverse sonde fluorescenti (DAPI, Cy3 e FITC).

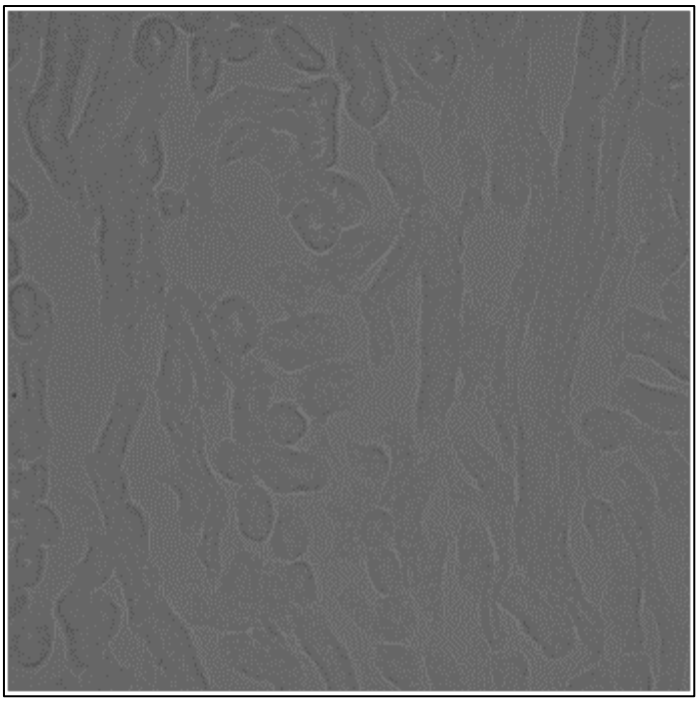


*Figura 1.1: sezione di rene di topo con ingrandimento 20x in modalità DIC.*

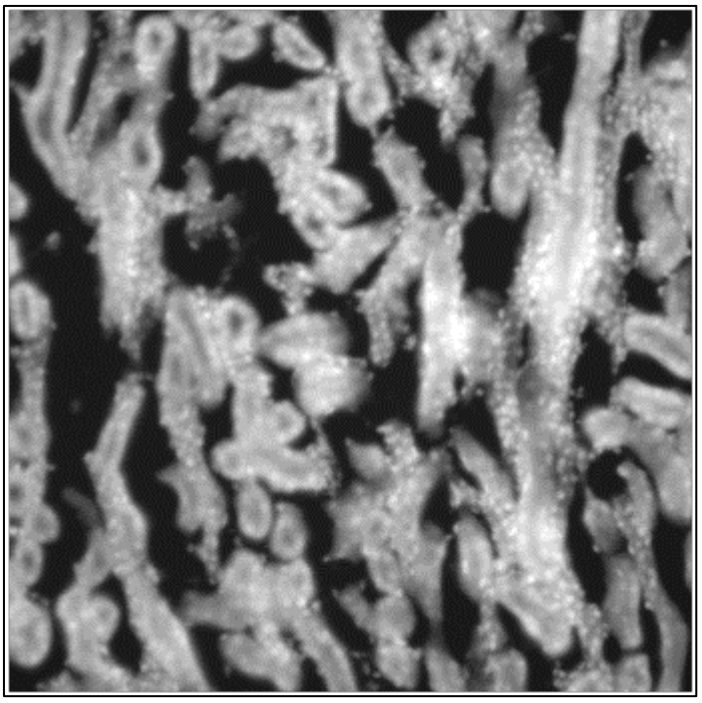


*Figura 1.2: sezione di rene di topo con ingrandimento 20x e colorante DAPI.*

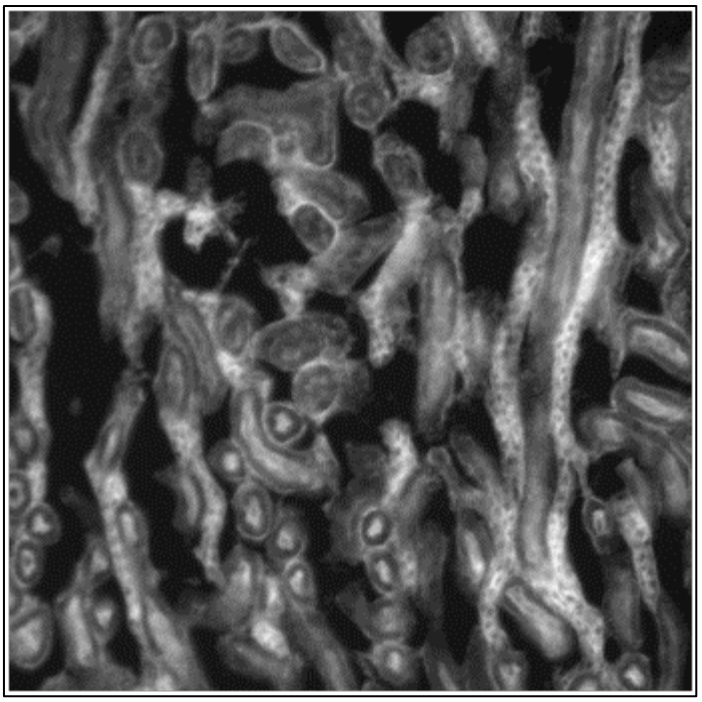


*Figura 1.3: sezione di rene di topo con ingrandimento 20x e colorante Cy3.*

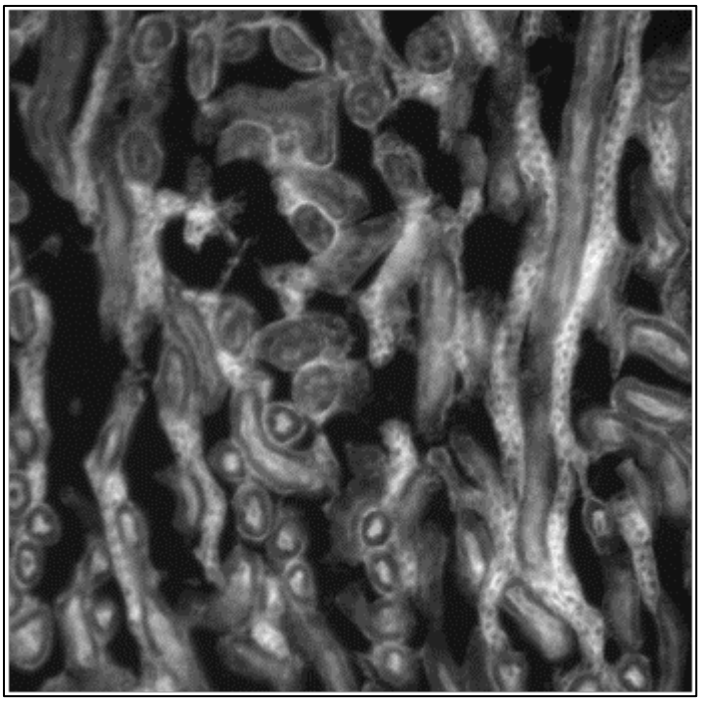


*Figura 1.4: sezione di rene di topo con ingrandimento 20x e colorante FITC.*

# 1 Tool per allineamento multimodale

Come menzionato in precedenza, si parla di *colocalizzazione* quando più immagini vengono allineate sulla base di una correlazione spaziale non casuale. Ciò significa che le immagini forniscono informazioni diverse sullo stesso oggetto in fase di studio e l'osservazione combinata di tali immagini può fornire una quantità maggiore di informazioni rispetto alle analisi condotte individualmente sui singoli segnali.

Quando si lavora con campioni istologici, è comune utilizzare coloranti al fine di evidenziare diverse parti subcellulari del campione. Tuttavia, per un'analisi più approfondita, spesso è necessario utilizzare più marcatori che non possono sempre essere applicati contemporaneamente e devono essere utilizzati in serie, con un lavaggio del campione tra l'applicazione di ciascun marcatore. Le immagini risultanti devono quindi essere allineate per consentire lo studio della colocalizzazione a livello di singola cellula, simulando una simultanea acquisizione dei vari segnali. Al momento, tuttavia, non esiste una procedura standard per l'allineamento delle immagini ottenute in questo modo.

## 1.1 Competitors

L'allineamento manuale richiede tempo ed è soggetto a potenziali errori. L'utilizzo di un software potrebbe rendere il processo più rapido e affidabile. Attualmente, esistono vari strumenti disponibili per la registrazione di immagini multimodali di microscopia 2D. In questa sezione, viene fornita una breve descrizione delle loro caratteristiche principali.

### 1.1.1 Align image by line ROI

*Align image by line ROI* (identificato con l'acronimo AIBLROI [3]) è un plugin popolare di *ImageJ/Fiji* sviluppato da Johannes Schindelin nel 2006 utilizzando Java. AIBLROI è estremamente semplice da utilizzare: l'utente deve semplicemente selezionare una linea fornendo due punti di riferimento per ciascuna immagine. L'ordine dei punti è importante, in quanto il primo punto nell'immagine sarà correlato al primo punto nella selezione della linea dell'altra immagine. Tuttavia, AIBLROI presenta alcune limitazioni:

(a) supporta l'allineamento di solo immagini in scala di grigi;

(b) funziona solo con due immagini alla volta;

(c) non fornisce parametri di output per riprodurre i risultati dell'allineamento.

### 1.1.2 BigWarp

*BigWarp* è un plugin per *ImageJ/Fiji* che permette l'allineamento elastico e manuale di immagini attraverso l'uso di punti di riferimento interattivi. È stato descritto nel 2016. Questo plugin consente agli utenti di definire manualmente le corrispondenze tra i punti di riferimento tramite un'interfaccia intuitiva, consentendo loro di posizionare coppie di punti di riferimento e visualizzare in tempo reale gli effetti della deformazione. Il modello di registrazione utilizzato da *BigWarp* si basa sull'utilizzo della tecnica *Thin Plate Spline*, che permette di definire una trasformazione deformabile uniforme che mappa in modo preciso i punti di riferimento. È possibile scegliere tra diversi modelli di registrazione, come il modello affine (trasformazione lineare con traslazione, rotazione, scale indipendenti e taglio), il modello di similarità (trasformazione lineare con traslazione, rotazione e un parametro di scala), il modello di rotazione (noto anche come modello rigido, una trasformazione lineare con traslazione e rotazione) o il modello di traslazione più semplice (solo spostamenti in X e Y). È inoltre possibile esportare e importare i punti di riferimento da file di testo. Una volta ottenuta l'immagine deformata, è possibile allinearla facilmente all'immagine di riferimento sfruttando le funzionalità offerte da *ImageJ/Fiji* (gli autori del plugin hanno fornito diversi video tutorial per mostrare come integrare in modo efficace *BigWarp* e *ImageJ/Fiji*) [4].

### 1.1.3 Correlia

*Correlia* è un plug-in open-source per *ImageJ/Fiji*, che funziona in modo indipendente dalla piattaforma e ha la capacità di gestire dati di microscopia 2D di vario tipo. È stato specificamente progettato per la co-registrazione di set di dati di microscopia multimodale 2D. *Correlia* è stato sviluppato presso "ProVIS - Center for Correlative Microscopy" (Helmholtz-Centre for Environmental Research, UFZ, Germania) ed è stato inizialmente concepito per soddisfare le esigenze della microscopia chimica, coinvolgendo diverse micrografie e mappe chimiche con diverse risoluzioni e campi visivi. Il plug-in è accompagnato da una documentazione esaustiva e da esempi di dati. Da un lato, il software offre diverse opzioni di registrazione manuale e automatica ed è integrato direttamente con *bUnwarpJ* (un altro popolare plug-in *ImageJ/Fiji*) per eseguire la co-registrazione elastica. D'altra parte, l'utilizzo di *Correlia* può essere complesso e richiede spesso la definizione di più parametri. Il plug-in offre diverse modalità di visualizzazione, ma le opzioni di esportazione sono limitate. Ad esempio, richiede la definizione di un frame di riferimento per

ritagliare tutte le immagini da registrare e non consente l'esportazione delle immagini allineate come uno stack multi-frame a piena risoluzione [5].

### 1.1.4 ec-CLEM

*ec-CLEM* è un software gratuito open source che funziona come un plugin nella piattaforma *Icy*. L'acronimo *ec-CLEM* significa "*easy cell-correlative light to electronic microscopy*" e riflette l'obiettivo del progetto: fornire uno strumento semplice per registrare immagini acquisite con microscopi ottici e microscopi elettronici. Attualmente, il software è in grado di gestire una vasta gamma di set di dati, inclusi immagini 2D e 3D (o una combinazione delle due dimensioni). *ec-CLEM* offre diverse opzioni per la registrazione manuale e automatica e supporta immagini time-lapse, multicanale e multidimensionali. La registrazione può essere rigida (applicando solo scala, rotazione e traslazione) o non rigida (con trasformazioni non lineari basate sull'interpolazione spline, dopo una trasformazione rigida iniziale) [6]. Il software valuta automaticamente la necessità di applicare deformazioni non rigide per ottenere una registrazione più accurata. *ec-CLEM* richiede l'installazione di *Icy* sul computer e potrebbe non essere immediato da utilizzare, ma rappresenta una soluzione completa e interessante per la co-registrazione delle immagini. Sono disponibili tutorial video e manuali di documentazione dettagliati sul sito web del software.

### 1.1.5 Elastix

*Elastix* è un software open source basato su *ITK* che fornisce una raccolta di algoritmi ampiamente utilizzati per risolvere problemi di registrazione delle immagini, principalmente nel campo medico. La sua architettura modulare consente agli utenti di configurare, testare e confrontare diversi metodi di registrazione per adattarli alle proprie esigenze specifiche. *Elastix* offre una varietà di opzioni, tra cui metodi di ottimizzazione, schemi multi-risoluzione, interpolatori, modelli di trasformazione e funzioni di costo. Il codice sorgente in linguaggio *C++* può essere compilato su diversi sistemi operativi, come Windows XP, Linux e Mac OS X. Mentre *Elastix* non ha una GUI ufficiale, esistono alcuni plugin che offrono un'interfaccia grafica per coloro che preferiscono utilizzare le funzionalità di *Elastix* in modo visuale. Ad esempio, *SlicerElastix* è un'estensione che integra *Elastix* all'interno del software *3D Slicer*. Inoltre, *Elastix* è supportato da *ITKElastix*, che rende il software disponibile in *Python*, e da *SimpleElastix*, che offre supporto per molti linguaggi di programmazione come *Java*, *R*, *Ruby*, *C#* e *Lua*. Gli autori hanno anche sviluppato una versione di *Elastix* per *ImageJ/Fiji*, sebbene l'installazione e la registrazione delle immagini con successo possano risultare complesse [7].

### 1.1.6 ITK

The *National Library of Medicine Insight Toolkit* (*ITK*) è un sistema open source multipiattaforma progettato per l'elaborazione di immagini mediche. Questo toolkit fornisce ai ricercatori nel campo dell'imaging medico una vasta gamma di algoritmi all'avanguardia per la registrazione, la segmentazione, l'analisi e la quantificazione dei dati medici. Inizialmente concepito nel 1999 per supportare il progetto specifico chiamato "*The Visible Human Project*", *ITK* si è evoluto nel corso degli anni diventando la tecnologia di base per molti prodotti commerciali di analisi dell'immagine medica in tutto il mondo. Nel 2005, la comunità *ITK* ha fondato *l'Insight Journal*, una rivista scientifica dedicata alla pratica del metodo scientifico. In questa rivista, tutti gli articoli devono includere la serie completa di codice sorgente, dati e parametri necessari per riprodurre le scoperte degli autori. Grazie all'*Insight Journal*, il repository *ITK* oggi contiene milioni di righe di codice sorgente ed è molto popolare. Tuttavia, va notato che *ITK* è stato progettato principalmente per scienziati e sviluppatori informatici, pertanto potrebbe risultare complesso per biologi o medici senza una formazione specifica in informatica. Per semplificarne l'utilizzo, gli autori hanno creato *SimpleITK*, un'interfaccia di programmazione semplificata che mette a disposizione gli algoritmi e le strutture dati di *ITK*. *SimpleITK* supporta diverse interfacce per diversi linguaggi di programmazione, andando oltre il *C++* originale. Tuttavia, al momento non esiste una GUI (interfaccia grafica) ufficiale per *ITK* o *SimpleITK*, quindi l'utilizzo di questi strumenti richiede competenze di programmazione [8].

### 1.1.7 Linear Stack Alignment with SIFT

*Linear Stack Alignment with SIFT* è un plugin *ImageJ/Fiji* sviluppato da Stephan Saalfeld nel 2008, ispirato dall'articolo scientifico di David Lowe che introduce il concetto di *SIFT* (Scale-Invariant Feature Transform). *Linear Stack Alignment with SIFT* è un algoritmo di registrazione completamente automatico che si basa su un'implementazione leggera di *SIFT* in Java. Per utilizzarlo, è necessario fornire alcune impostazioni e uno stack di immagini in input. Come risultato, il plugin restituisce un nuovo stack di immagini allineate, con la prima immagine del set considerata come riferimento [9].

### 1.1.8 Register Virtual Stack Slices

*Register Virtual Stack Slices* (identificato con l'acronimo *RVSS*) è un plugin per *ImageJ/Fiji* creato dagli stessi autori di *bUnwarpJ*. *RVSS* è stato sviluppato per la co-registrazione di una sequenza di sezioni di immagini che sono memorizzate in una cartella, creando un nuovo insieme di sezioni di immagini registrate con una risoluzione migliore. *RVSS* offre sei tecniche di registrazione preselezionate:

(a) Traduzione (solo spostamenti in X, Y);

(b) Rigido (traslazione e rotazione);

(c) Somiglianza (traslazione, rotazione e ridimensionamento isotropico);

(d) Affine (traslazione, rotazione, ridimensionamento e taglio);

(e) Spostamento dei minimi quadrati;

(f) Elastico (tramite *bUnwarpJ*).

Tutti i modelli si avvalgono delle funzioni *SIFT* che vengono estratte automaticamente.
*RVSS* ha un'interfaccia utente (GUI) semplice, ma offre anche opzioni avanzate di configurazione tramite caselle di controllo per diverse opzioni di registrazione, che sono le stesse offerte da Linear *Stack Alignment with SIFT*. Inoltre, il plugin è integrato in modo più efficiente con *bUnwarpJ*. *RVSS* consente anche di salvare le trasformazioni risultanti in file ".xml" [10].

### 1.1.9 StackReg

Il plug-in *ImageJ/Fiji* chiamato *StackReg* è stato sviluppato per l'allineamento ricorsivo di una pila di immagini. Inizialmente progettato per registrare una pila di fette dello stesso campione, *StackReg* è ottimizzato per immagini acquisite con la stessa modalità di imaging ma relative a sezioni a profondità diverse. Tuttavia, non è specificamente ottimizzato per la registrazione di immagini multimodali in cui le modalità di imaging possono variare. *StackReg* utilizza ciascuna fetta come modello di riferimento per allineare la fetta successiva, consentendo così un allineamento basato sulla propagazione. Tuttavia, va notato che il plug-in *StackReg* richiede l'installazione di un secondo plug-in chiamato *TurboReg* per funzionare correttamente. *StackReg* offre cinque tipi di modelli di registrazione, ma nessuno di essi considera le trasformazioni elastiche, che possono essere utili per la registrazione di immagini che richiedono una deformazione non lineare per un allineamento accurato [11].

### 1.1.10 TrakEM2

*TrakEM2* è un plug-in *ImageJ/Fiji* che offre molteplici funzionalità per il data mining morfologico, la modellazione tridimensionale, lo stitching di immagini, la registrazione, l'editing e l'annotazione. Una delle sue principali caratteristiche è la registrazione di riquadri di immagini mobili utilizzando gli algoritmi *SIFT* e di ottimizzazione globale. *TrakEM2* supporta diverse modalità di registrazione, tra cui la registrazione manuale, semiautomatica e completamente automatica delle immagini. È possibile eseguire la registrazione all'interno delle sezioni o tra le sezioni utilizzando modelli di registrazione come traduzione, rigido, somiglianza e affine. Inoltre, *TrakEM2* fornisce anche un algoritmo per l'allineamento elastico che tiene conto delle distorsioni non lineari per ottenere un allineamento accurato. Il plug-in consente anche di salvare le trasformazioni risultanti in file ".xml" e offre funzionalità per salvare e caricare i progetti. Sebbene *TrakEM2* offra un'ampia gamma di funzionalità, viene fornito con manuali dettagliati e tutorial video per supportare i ricercatori nell'utilizzo del software [12].

### 1.1.11 Elastic Alignment e Montage

I plugin "*Elastic Alignment*" e "*Montage*" sono funzionalità incorporate nel software "*TrakEM2*". Questi plugin sono utilizzati quando si lavora con ampie serie di sezioni multi-tile tramite il menu di allineamento del software. Il plugin "*Elastic Alignment*" si occupa di allineare serie di sezioni deformate, mentre il plugin "*Montage*" crea mosaici combinando immagini sovrapposte che presentano deformazioni non lineari. L'obiettivo è deformare le immagini in modo da ottenere sovrapposizioni ottimali tra di esse. Ogni deformazione viene calcolata in modo da ridurre al minimo la deformazione complessiva del mosaico finale. Entrambi i plugin funzionano esclusivamente con stacks di immagini, consentendo un'elaborazione coerente e integrata delle immagini all'interno di *TrakEM2* [13].

### 1.1.12 Moving Least Squares

Il plugin *Moving Least Squares* utilizza il metodo dei minimi quadrati ponderati. Questa tecnica è utile per deformare un'immagine singola utilizzando una serie di punti di riferimento. Tuttavia, non si occupa dell'allineamento tra immagini. A differenza del plugin *bUnwarpJ*, che utilizza deformazioni elastiche, le trasformazioni implementate nel plugin *Moving Least Squares* sono considerate rigide. Gli autori del plugin hanno voluto dimostrare che è possibile ottenere deformazioni utilizzando trasformazioni geometricamente simili, ovvero combinando

trasformazioni rigide con ridimensionamenti isometrici. Questo approccio permette di evitare la complessità delle minimizzazioni non lineari [14].

### 1.1.13 Image Stitching

Il plugin "*Image Stitching*" richiede come input una griglia quadrata di immagini. Utilizzando il Teorema di Fourier, calcola tutte le possibili traslazioni tra le immagini inserite contemporaneamente. In output fornisce la migliore sovrapposizione tra le immagini utilizzando misure di cross correlazione. Questo plugin è in grado di allineare un numero arbitrario di canali di natura diversa, rendendolo multimodale. Per garantire una transizione uniforme tra le porzioni della griglia e rimuovere le differenze di luminosità, viene applicata una correzione d'intensità non lineare sui bordi delle immagini. Per il stitching tra due immagini, è possibile utilizzare la modalità "Pairwise Stitching", in cui vengono fornite immagini con punti di interesse definiti per calcolare le matrici di correlazione di fase. La modalità "Grid/Collection Stitching" consente di organizzare le immagini in una griglia rispetto a tutte le possibili orientazioni, ad esempio riga per riga o colonna per colonna [15].

### 1.1.14 BigStitcher

"*BigStitcher*" è la versione successiva di "*Image Stitching*" [16] e mantiene le stesse funzionalità. Tuttavia, presenta un importante miglioramento: è in grado di elaborare e visualizzare immagini di qualsiasi dimensione, grazie alla sua invarianza di dimensione. Attualmente, il software è in grado di effettuare un'ottimizzazione globale che permette di allineare insiemi di dati con connessioni sparse, dove il contenuto dell'immagine è separato da aree in cui lo sfondo è quasi costante. Tuttavia, è importante notare che il software si trova ancora nella fase di beta test e non è considerato completo. Oltre a queste funzionalità, "*BigStitcher*" offre anche altre caratteristiche, tra cui:

- 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 le 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.

### 1.1.15 MIST

"*MIST*" è un algoritmo di stitching per collezioni di griglie di immagini 2D, che sta per "*Microscopy Image Stitching Tool*". Gli sviluppatori hanno creato un metodo per migliorare il processo di Phase-Correlation utilizzando la Trasformata di Fourier. Questo metodo ottimizza il calcolo delle traslazioni utilizzando l'algoritmo "Hill Climbing", limitando il numero di iterazioni a quattro. Questo approccio è particolarmente adatto per immagini sparse di grandi dimensioni con set di dati di riferimento utilizzati per calcolare gli errori di assemblaggio nel mosaico finale. È importante notare che questo software è disponibile solo per *ImageJ/Fiji* a 64 bit [17].

### 1.1.16 MicroMos

*MicroMos* è un software open-source che consente di creare automaticamente un mosaico di immagini unendo immagini parzialmente sovrapposte. È importante notare che le immagini fornite in input devono essere statiche e non deformabili. Durante la fase di stitching, vengono applicati algoritmi per correggere l'effetto di vignettatura, ovvero la diminuzione dell'intensità o la saturazione dell'immagine che può verificarsi ai bordi. Questo software può essere utilizzato per la registrazione di immagini multi-modalità, tuttavia, l'utente deve definire manualmente la posizione di sovrapposizione delle immagini [18].

## 1.2 Tabella comparativa plugin/tools

| **Nome** | **Link** | **Dataset disponibile [Y/N]** | **Tutorial disponibile [Y/N]** | **Gestione scale differenti [Y/N]** | **Gestione size differenti [Y/N]** | **Allinea. Manuale [Y/N]** | **Allinea. Automatico [Y/N]** | **Allinea. Automatico multimodale [Y/N]** |
|---|---|---|---|---|---|---|---|---|
| *Align images by ROI* | *Align images by ROI* | N | Y | Y | N | N | Y | Y |
| *BigWarp* | *BigWarp* | N | Y | Y | Y | Y | Y | N |
| *Correlia* | *Correlia* | Y | Y | Y | Y | Y | Y | Y |
| *ec-CLEM* | *ec-CLEM* | Y | Y | Y | Y | Y | Y | Y |
| *Elastix* | *Elastix* | N | Y | Y | Y | N | Y | Y |
| *ITK* | *ITK* | Y | Y | Y | Y | Y | Y | Y |
| *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 |
| *StackReg* | *StackReg* | Y | N | Y | N | Y | Y | Y |
| *TrackEM2* | *TrackEM2* | Y | Y | Y | Y | Y | Y | Y |
| *Elastic Alignment e Montage* | *Elastic Alignment e Montage* | N | Y | Y | Y | N | N | Y |
| *Moving Least Squares* | *Moving Least Squares* | N | Y | N | N | N | Y | N |
| *Image Stitching* | *Image Stitching* | N | Y | N | Y | N | Y | Y |
| *BigStitcher* | *BigStitcher* | Y | Y | N | Y | Y | Y | Y |
| *MIST* | *MIST* | Y | Y | N | N | N | Y | N |
| *MicroMos* | *MicroMos* | Y | Y | N | N | Y | Y | N |

Nella tabella sopra riportata sono state riassunte le feature principali dei vari tool competitor disponibili gratuitamente nel web. Con il termine “Y” si fa riferimento alla “presenza” della feature in esame; “N” altrimenti.

## 1.3 Limiti principali

Lo studio della letteratura fatto per individuare i competitors ha evidenziato la presenza di sei competitors principali:

- *BigWarp;*
- *Correlia;*

- *ec-CLAM;*
- *Elastix;*
- *TrackEM2;*
- *ITK*.

Per ognuno dei *software* qui sopra elencati si definiscono i seguenti limiti qui a seguito descritti:

### 1.3.1 BigWarp

*BigWarp* ha alcune limitazioni che includono la necessità di un allineamento completamente manuale e l'uso di un computer moderno per sfruttarne appieno le funzionalità. Il termine "Big" nel suo nome si riferisce al concetto di Big Data, e una delle sue caratteristiche principali consiste proprio nell'operare con dataset di dimensioni considerevoli. Inoltre, è importante sottolineare che il plugin appena descritto non dispone di strumenti per le deformazioni elastiche e non è in grado di preservare le caratteristiche originali delle immagini, come la profondità di ogni pixel e le eventuali LUT.

### 1.3.2 Correlia

*Correlia* ha alcune limitazioni significative da considerare: come la maggior parte dei competitor elencati, richiede all'utente di specificare le trasformazioni da applicare durante la fase di allineamento automatico, il che rende il processo particolarmente lento. Altri aspetti negativi da tenere in considerazione includono l'uso di algoritmi obsoleti e la dipendenza da *bUnwarpj*. *Correlia* presenta anche altre limitazioni, come l'incapacità di gestire immagini multicanale e il fatto che ogni volta che viene eseguita un'operazione di allineamento, le immagini di output perdono le proprietà originali, come la profondità di ogni pixel o eventuali LUT. Inoltre, la dimensione di ogni immagine viene impostata a una grandezza prefissata a priori, con il rischio di perdere la qualità dell'immagine originale.

### 1.3.3 ec-CLAM

Il principale inconveniente di *ec-CLEM* è rappresentato dal grande numero di finestre di dialogo richieste per utilizzare l'*Autofinder*, il che può portare facilmente a confusione nell'ordine degli eventi e risultare scomodo per gli utenti con competenze medie nell'analisi dei dati e nell'uso di applicazioni. Inoltre, è importante notare che *ec-CLEM* non dispone di strumenti per eseguire

deformazioni elastiche, né è in grado di gestire immagini multicanale o preservare le proprietà originali delle immagini di partenza.

### 1.3.4 Elastix

*Elastix* presenta sicuramente come svantaggio la sua complessità d'uso. La procedura di installazione può essere soggetta a errori in quanto richiede il download sia del plugin tramite l'updater di *Fiji*, che comprende solo la GUI, sia del vero e proprio programma dal repository di *GitHub*, con la necessità di specificare il percorso della cartella durante l'esecuzione. Inoltre, l'elenco dei parametri è lungo e poco intuitivo per gli utenti. *Elastix*, come molti altri, non supporta le deformazioni elastiche, non è in grado di gestire immagini multicanale e non conserva le proprietà originali dell'immagine dopo aver eseguito le operazioni di allineamento.

### 1.3.5 TrackEM2

*TrackEM2* è principalmente progettato per analizzare immagini di grandi dimensioni e offre la possibilità di impostare vari flag. Questo è un primo segnale della sua complessità d'uso, in quanto richiede una certa conoscenza informatica da parte dell'utente per sfruttarlo appieno. L'applicazione non è originariamente concepita per la registrazione, ma per l'estrazione di dati morfologici. Il suo utilizzo principale consiste nella misurazione di volumi, superfici e lunghezze utilizzando punti di riferimento. L'uso dell'algoritmo *SIFT* per la registrazione comporta un ulteriore costo computazionale. Sebbene l'allineamento automatico tramite *SIFT* sia possibile, richiede una serie di passaggi attraverso finestre di dialogo per la configurazione. Questi aspetti impediscono un utilizzo rapido e, nonostante sia un ottimo software, risulta macchinoso. Inoltre, *TrackEM2* presenta altre limitazioni, come l'incapacità di eseguire deformazioni elastiche e la mancanza di supporto per le proprietà originali delle immagini dopo che queste sono state allineate.

### 1.3.6 ITK

Uno dei principali difetti di *ITK* è rappresentato dalla sua complessità d'uso. Questo toolkit richiede una conoscenza approfondita della programmazione e una buona comprensione dei concetti matematici sottostanti. Le sue molteplici funzionalità e le numerose opzioni di configurazione possono risultare intimidatorie per gli utenti meno esperti, rendendo necessario un lungo periodo di apprendimento per utilizzarlo efficacemente. Un altro difetto di *ITK* è la configurazione complessa.

L'utente deve affrontare una serie di parametri di allineamento e opzioni di configurazione che possono risultare poco intuitivi. La scelta delle impostazioni corrette richiede una conoscenza approfondita delle tecniche di allineamento e un'attenta sperimentazione, rendendo il processo di configurazione laborioso e potenzialmente suscettibile a errori.

## 1.4 DS4H Image Alignment tool

In questo capitolo viene presentata la versione più recente del plugin *DS4H Image Alignment* che, grazie ai 3 moduli sviluppati descritti nell'introduzione: (*a*) include una funzionalità di allineamento automatico ottimizzata e completamente funzionante; (*b*) include l'utilizzo del plugin *bUnwarpJ*; (c) supporta immagini RGB e multicanale in generale. Queste caratteristiche verranno analizzate più approfonditamente nei capitoli successivi.

### 1.4.1 Presentazione generale

Il plugin presenta un'interfaccia grafica composta di finestre modali mostrate a seconda della funzionalità richiesta. Le finestre più informative sono:

- *Finestra Principale* - rappresenta l'area in cui vengono visualizzate tutte le immagini in uso, oltre a consentire la visualizzazione delle immagini, offre diverse funzionalità, tra cui la modifica degli algoritmi di allineamento automatico e manuale, la configurazione dei parametri per il plugin *bUnwarpJ*, la gestione dell'intero progetto di lavoro tramite le opzioni di "*import*", "*export*" o "*clear*", e la possibilità di selezionare il target tra tutte le immagini e rimuovere eventuali immagini non necessarie;
- *Finestra di preview* – consente di visualizzare un'immagine singola e offre la possibilità di aggiungere o rimuovere punti per l'allineamento e inoltre gestire la dimensione ed il colore dei vari punti specificati;
- *Finestra di output* – visualizza il risultato finale dell'allineamento, offrendo la possibilità di salvare le immagini, riutilizzarle per un successivo allineamento o applicare l'algoritmo di deformazione elastica *bUnwarpJ* a tutte le immagini.

## 1.5 Modalità d'uso dell'applicazione DS4H

Di seguito vengono illustrate tutte le funzionalità disponibili per gli utenti. Le schermate mostrate in questa sezione sono state catturate utilizzando un computer Lenovo con il sistema operativo *Ubuntu*.

### 1.5.1 Caricamento delle immagini

Dopo il lancio del nostro plugin, la prima finestra che si aprirà sarà la finestra principale (Figura 1.1), è possibile osservare la struttura di questa finestra. Per effettuare il caricamento delle immagini si può fare affidamento a due voci presenti nel menu in alto:

- "File";
- "Project".

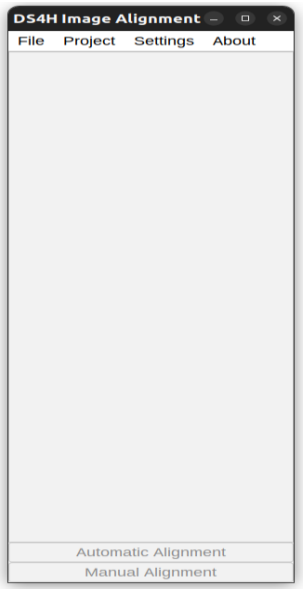



*Figura 1.1: Finestra principale.*

### 1.5.2 Caricamento delle immagini tramite voce File

Dopo aver selezionato la voce "File" nella barra dei menu, verrà aperta un'altra finestra per il caricamento delle immagini (Figura 1.2), sarà possibile esplorare le cartelle del sistema e selezionare un gruppo di immagini da caricare all'interno del plugin.

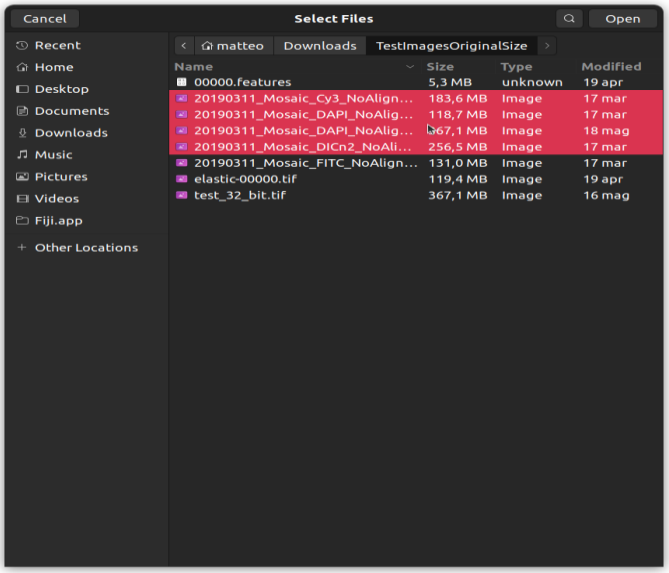



*Figura 1.2: Selezione multipla di immagini tramite "File".*

### 1.5.3 Caricamento delle immagini tramite voce Project

In seguito ad aver cliccato su "Project", verrà visualizzato un sottomenu dal quale sarà necessario selezionare "Import". Successivamente, si aprirà una finestra (Figura 1.3), dalla quale sarà possibile scegliere una cartella. Tutte le immagini presenti nella cartella selezionata verranno importate automaticamente.

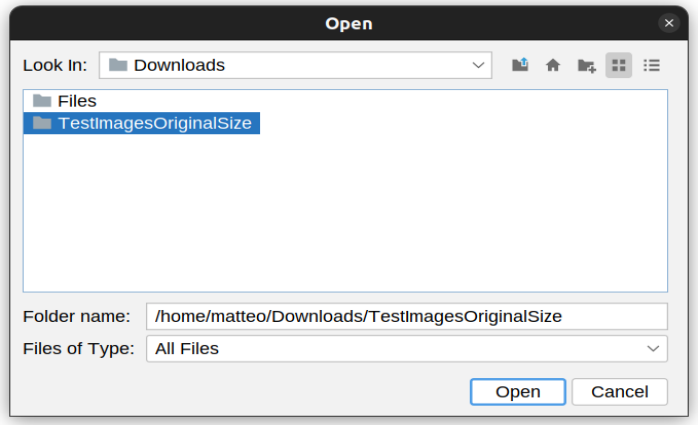



*Figura 1.3: Caricamento di immagini tramite "Project > Import".*

### 1.5.4 Impostazioni del Plugin

Nel nostro software, diamo anche all'utente la possibilità di personalizzare le impostazioni degli algoritmi automatici, manuali e di deformazione elastica. Se si clicca sulla voce "Settings" nella barra dei menu, si aprirà un sottomenu con tre opzioni:

1. "*Manual*" - selezionando questa voce, il menù dell'allineamento manuale si aprirà (Figura 1.4) dando possibilità all'utente di configurare:
    - "Algorithm", per selezionare il tipo di algoritmo di allineamento manuale che si andrà ad utilizzare, varia tra "*Traslativo*", "*Affine*" e "*Proiettivo*";
    - si potrà scegliere se applicare "*Traslazione*", "*Scala*" e "*Rotazione*";
    - "Point Overload", selezionare come gestire eventuale abbondanza di punti nel caso in cui questi non fossero necessari, selezionando "*First point available*", "*RANSAC*" o "*Minimum Least Square"*.

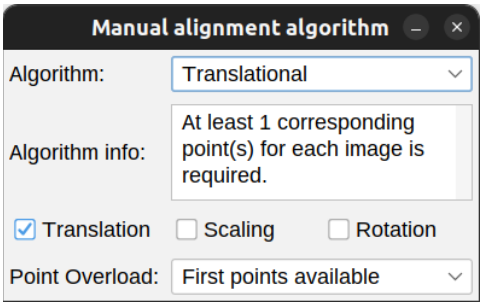



*Figura 1.4: Menù di configurazione algoritmo manuale.*

2. "*Automatic* "– cliccando su tale voce, si aprirà un menu (Figura 1.5) in cui è possibile impostare:
    - "*Detector*", il tipo di algoritmo automatico per individuare automaticamente i punti per l'allineamento;
    - "*Threshold factor*", per selezionare anche eventuali punti "non ottimi":
    - "*Scaling factor*", ci permette di selezionare di quante volte ogni singola immagine va scalata;
    - "*Algorithm*", quale algoritmo si andrà ad utilizzare una volta che si saranno individuati i punti automaticamente, questi algoritmi sono gli stessi del metodo manuale;
    - Se applicare "*Traslazione*", "*Scala*" e "*Rotazione*".

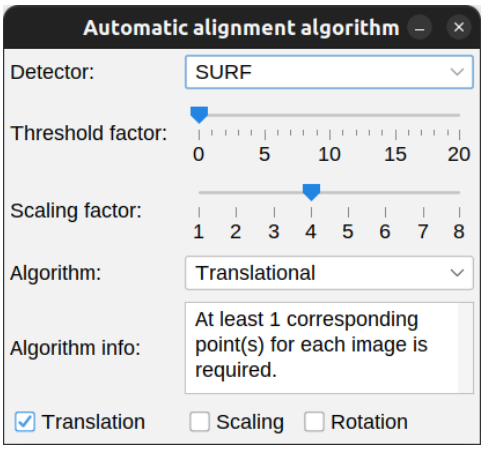



*Figura 1.5: Menù di configurazione algoritmo automatico.*

3. "*bUnwarpJ*": - selezionando questa voce, sarà possibile configurare i parametri da utilizzare durante l'algoritmo di deformazione elastica (Figura 1.6), tra i tanti parametri che si possono impostare mi soffermo sul più rilevante, "Mode", in particolare questo ci permette di scegliere tra tre diverse voci:
    - "*Mono*", fa sì che il programma esegua solo una registrazione unidirezionale, cioè dallo sorgente alla destinazione
    - "*Accurate*" e "*Fast*", prevedono la registrazione bidirezionale e influiscono sui criteri di arresto utilizzati internamente dal programma.

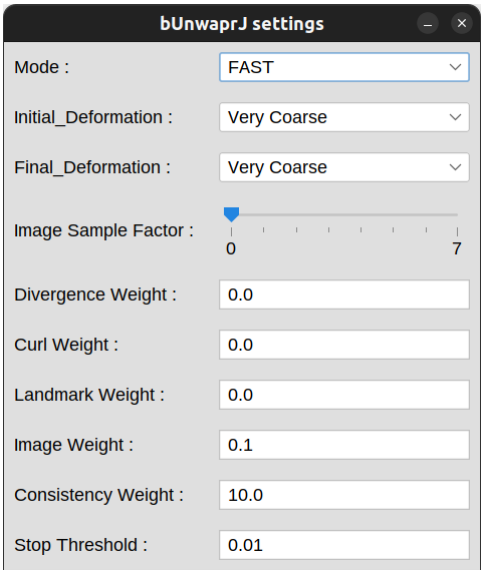



*Figura 1.6: Menù di configurazione bUnwarpJ.*

### 1.5.5 Scelta immagine target

Dopo aver caricato correttamente le immagini nel nostro plugin, per eseguire l'allineamento manuale ed automatico è necessario selezionare tra tutte le immagini caricate, quale sia l'immagine "Target". Questa immagine rappresenta la base sulla quale gli allineamenti dovranno essere eseguiti, più nello specifico tutte le immagini verranno alterate in base a questa immagine. Per impostare il "Target" basterà semplicemente cliccare sul bottone "TARGET" (Figura 1.7).

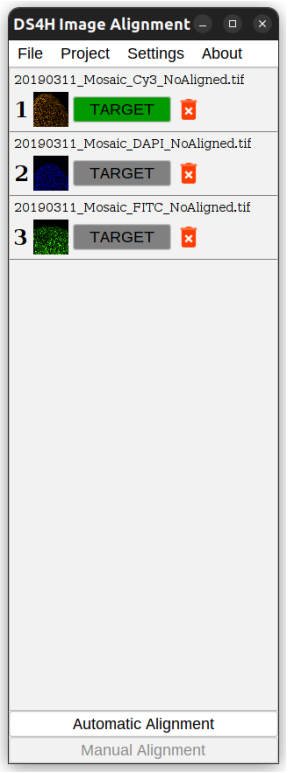



*Figura 1.7: Finestra principale con immagini caricate.*

### 1.5.6 Rimozione Immagine

Nel caso in cui dopo aver caricato correttamente le immagini nel nostro plugin, l'utente nota di aver caricato inconsapevolmente un'immagine non necessaria, potrà rimuoverla dal progetto cliccando sull'icona del cestino rosso vicino al bottone "TARGET" (Figura 1.7). In questo modo l'immagine verrà rimossa completamente dal progetto.

### 1.5.7 Gestione corner points

Dopo aver caricato correttamente le immagini nel nostro plugin, per eseguire l'allineamento manuale è necessario indicare lo stesso numero di corner points su ciascuna immagine caricata. Il numero minimo di punti richiesti varia a seconda dell'algoritmo selezionato. Per *impostare* i corner point su ciascuna immagine, sarà necessario fare clic sull'immagine desiderata nella finestra principale. Successivamente, si aprirà la *finestra di* preview (Figura 1.8), in cui sarà possibile lavorare sull'immagine selezionata.

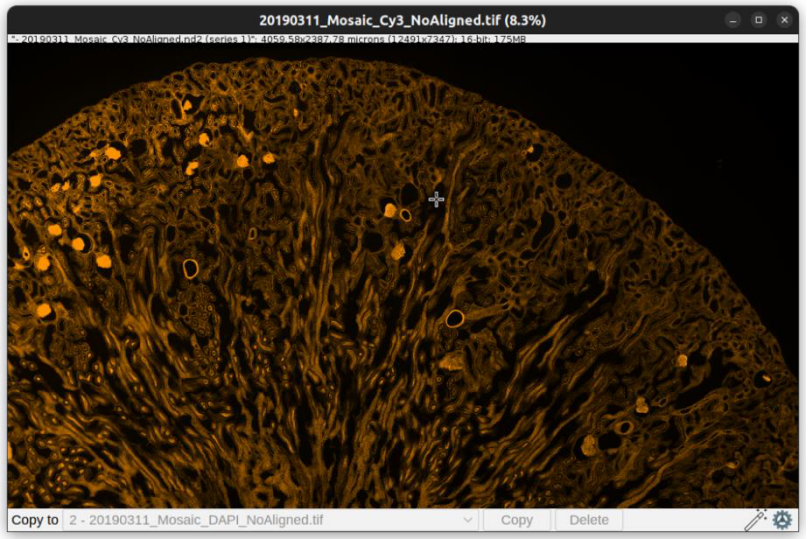



*Figura 1.8: Finestra di preview.*

### 1.5.8 Aggiunta dei corner points

Se si preme due volte sul tasto sinistro del mouse all'interno dei limiti dell'immagine corrente, verrà creato un *corner point* nel punto esatto in cui è posizionato il puntatore (Figura 1.9). Si può vedere il risultato di questa operazione, con la creazione di tre corner points. L'ordine di creazione dei punti è evidenziato grazie alla numerazione automatica.

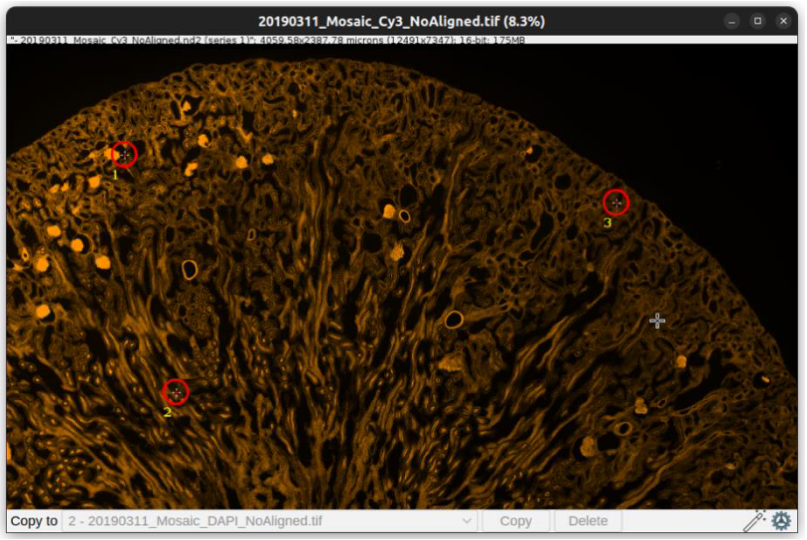



*Figura 1.9: Finestra di preview in cui sono stati aggiunti 3 punti.*

### 1.5.9 Selezione singola o multipla dei corner points

Come detto precedentemente è possibile inserire anche più corner all'interno della stessa immagine. È possibile selezionarli singolarmente (Figura 1.10) o in modo multiplo (Figura 1.11). Quando si selezionano più corner, è possibile spostarli tutti insieme, lo spostamento dei corner points è limitato nei confini dell'immagine, così facendo l'utente non potrà inserire i corner points all'esterno dei confini dell'immagine su cui si sta lavorando.

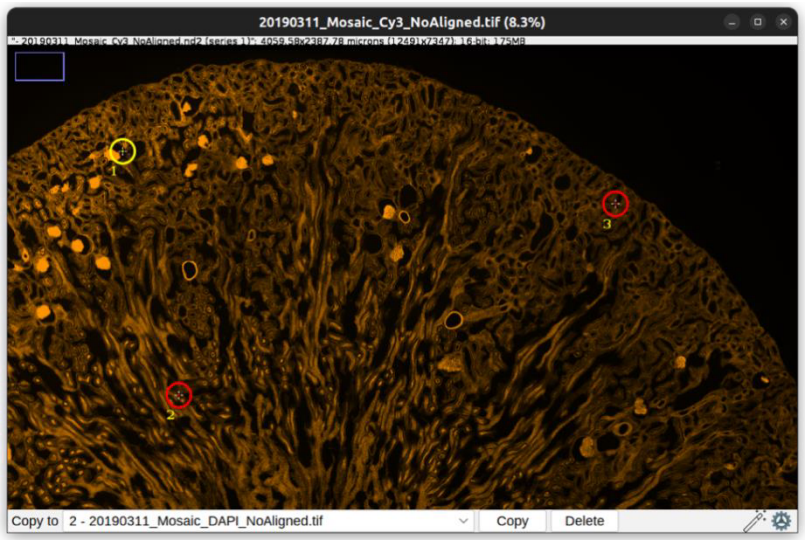



*Figura 1.10: Esempio selezione singola.*

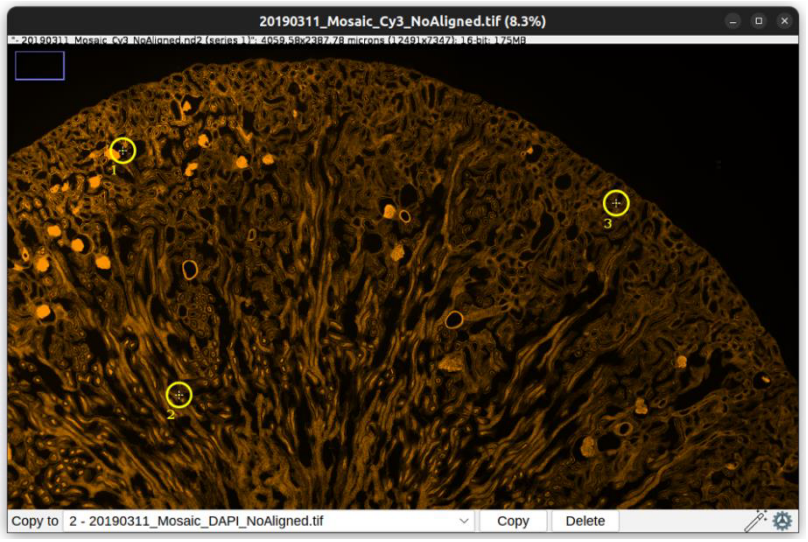



*Figura 1.11: Esempio selezione multipla.*

### 1.5.10 Copia dei corner points

Al fine di facilitare l'uso del nostro plugin, abbiamo implementato anche la funzionalità di copiare i corner points inseriti su un'immagine e incollarli su un'altra. Per iniziare, è necessario selezionare almeno un corner point all'interno dell'immagine. Successivamente, sarà possibile scegliere l'immagine di destinazione in cui si desidera incollare i corner points, selezionandola dal menù a tendina nella voce "*copy to*" (Figura 1.12). Infine, cliccando sul pulsante "*Copy*", l'operazione verrà completata. Nel caso in cui alcuni dei corner point copiati dovessero finire al di fuori dell'immagine, verrà notificato all'utente (come mostrato nella Figura 1.13) e il corner point non verrà inserito nell'immagine di destinazione.

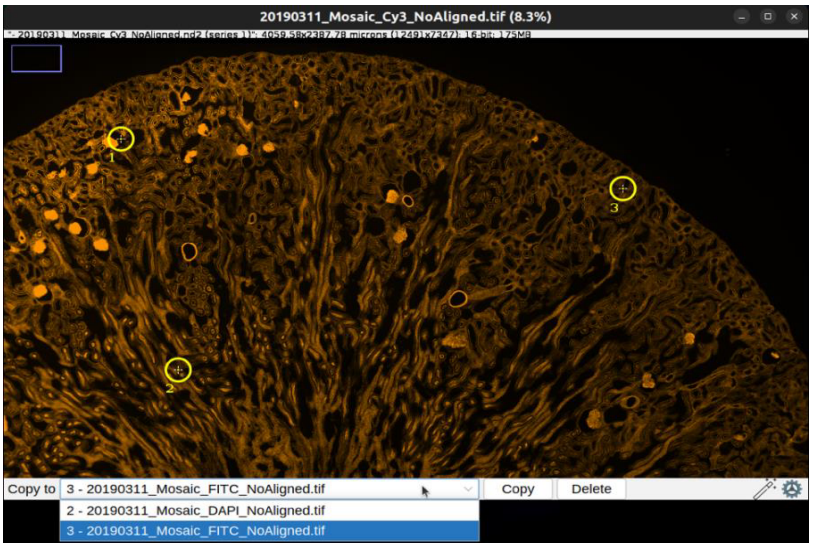



*Figura 1.12: Esempio di selezione copia di corner point.*

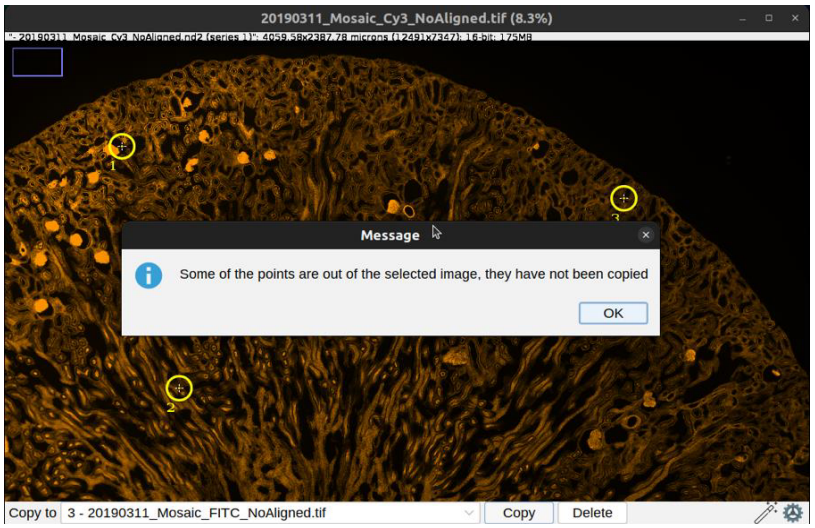



*Figura 1.13: Esempio di copia di corner point fuori dall'immagine.*

### 1.5.11 Rimozione dei corner points

Dopo aver selezionato uno o più corner points, è possibile rimuoverli completamente dall'immagine premendo il pulsante "Delete" (Figura 1.14) con il cursore. Ciò comporterà la completa eliminazione dei corner point selezionati dall'immagine (Figura 1.15) e automaticamente gli indici dei corner points verranno scalati per adattarsi alla nuova configurazione.

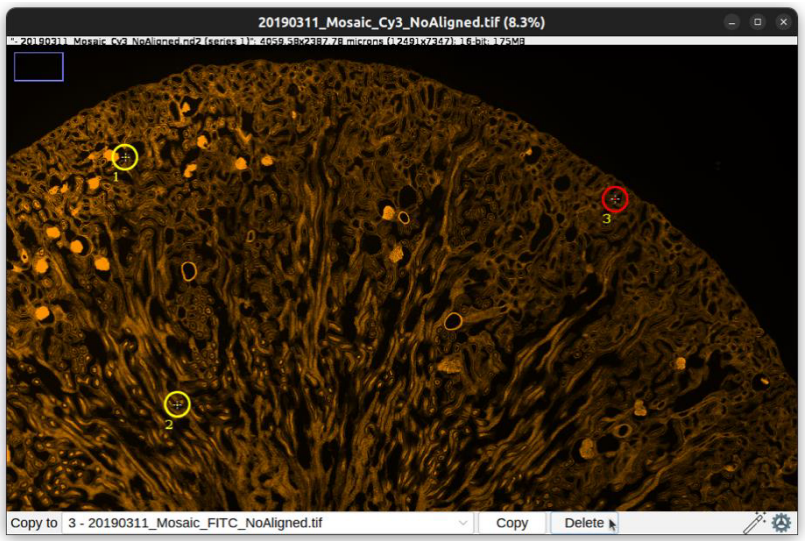



*Figura 1.14: Esempio selezione corner point 1 e 2 per la rimozione.*

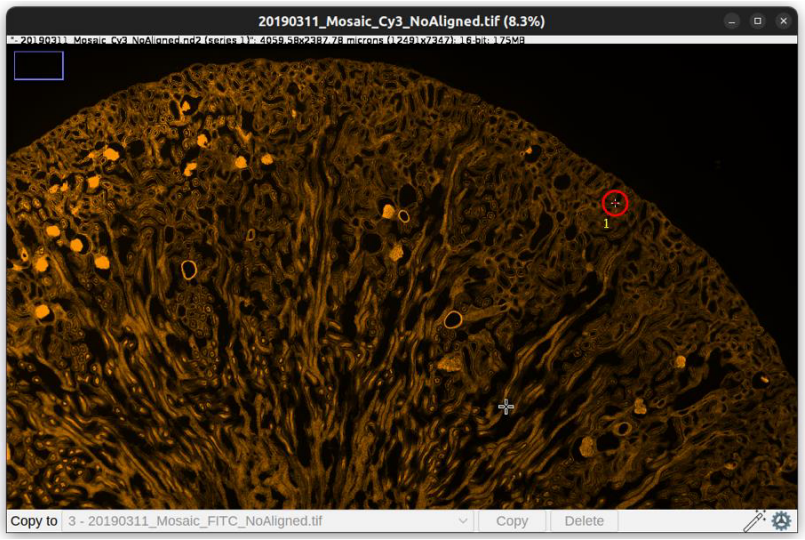



*Figura 1.15: Risultato dopo aver eliminato i punti 1 e 2.*

### 1.5.12 Modifica dei corner points

Nel caso in cui il colore dei corner points produce un contrasto poco significativo o se si desidera modificare l'ordine tra due corner points inseriti, è possibile apportare modifiche sia all'aspetto e sia all'ordine di ciascun corner point. Ciò può essere fatto facendo clic sull'icona delle impostazioni, che aprirà la finestra delle impostazioni per i corner points inseriti (Figura 1.16). Attraverso questa finestra è possibile:

- Variare il colore del corner;
- Cambiare il colore di selezione del corner;
- Alterare il colore dell'indice del corner;
- Modificare la dimensione del corner;
- Sostituire il numero dell'indice di un corner point con quello di un altro corner inserito

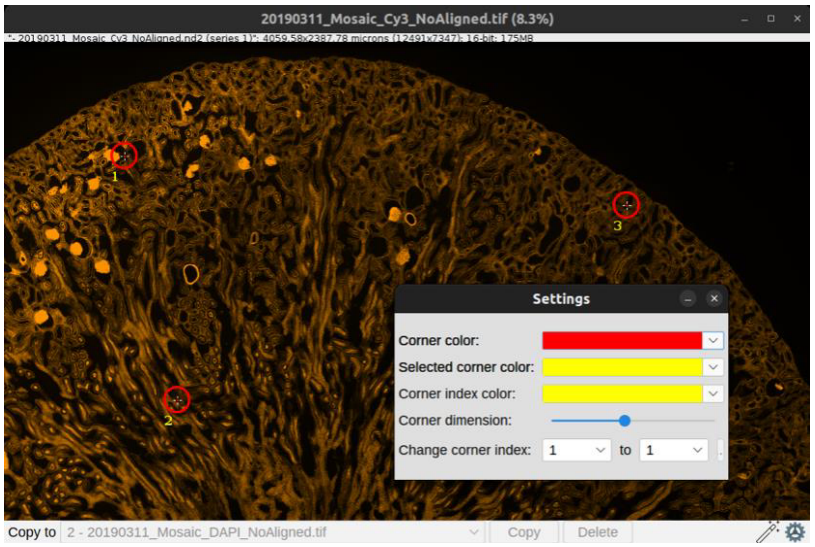



*Figura 1.16: Finestra di settings per i punti impostati.*

### 1.5.13 Esportare il progetto

Una volta che l'utente ha caricato le immagini, può eseguire l'operazione di "Project > Export" per esportare l'intero progetto di lavoro. In particolare, utilizzando questa funzione, l'utente può salvare tutte le immagini con i rispettivi punti impostati su ciascuna immagine, oltre al target selezionato, in una qualsiasi cartella del proprio sistema. Nel momento in cui si clicca la voce “Export”, verrà chiesto all’utente (Figura 1.17) in che percorso effettuare il salvataggio del progetto, automaticamente, verrà creata una cartella chiamata "*YYYYMMDD_HHMM_DS4H_Project*" (con *YYYYMMDD* indicante l’anno il mese e il giorno del salvataggio, *HHMM* indicante l’ora ed il minuto) contente tutte le immagini e un singolo file in formato JSON contenente tutte le informazioni sul “target” e sui corner points.

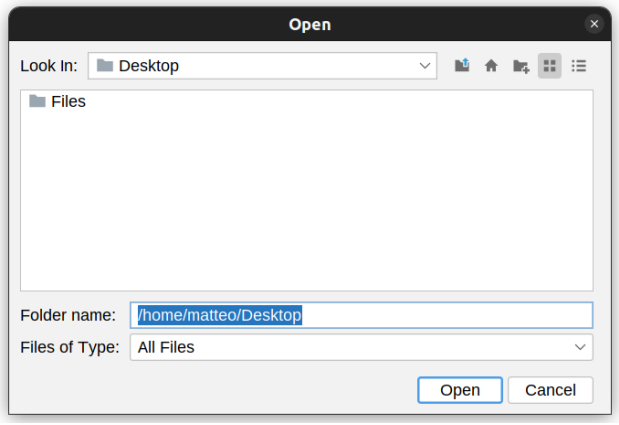



*Figura 1.17: Esempio di export del progetto nella cartella Desktop.*

### 1.5.14 Allineamento manuale

Una volta che su ogni immagine sono stati specificati i corner point, sarà possibile allinearle cliccando sul tasto “Manual Alignment” presente nella *finestra principale.* Ora che l’allineamento è partito verrà visualizzata una finestra che ci permetterà di capire a che punto siamo con il nostro allineamento (Figura 1.18).

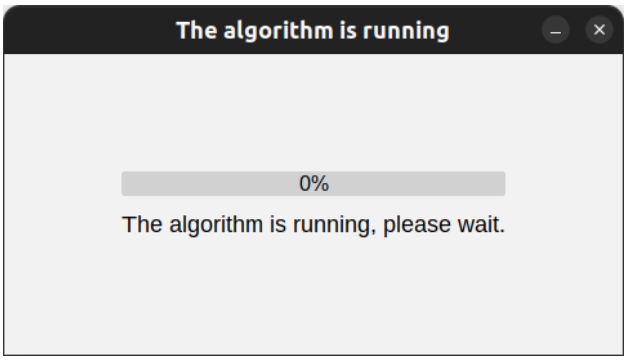



*Figura 1.18: Finestra di loading.*

Quando l’allineamento sarà terminato, verrà visualizzata la *finestra di output*, la quale contiene uno stack scorribile con tutte le immagini allineate in base al target selezionato (Figura 1.19).

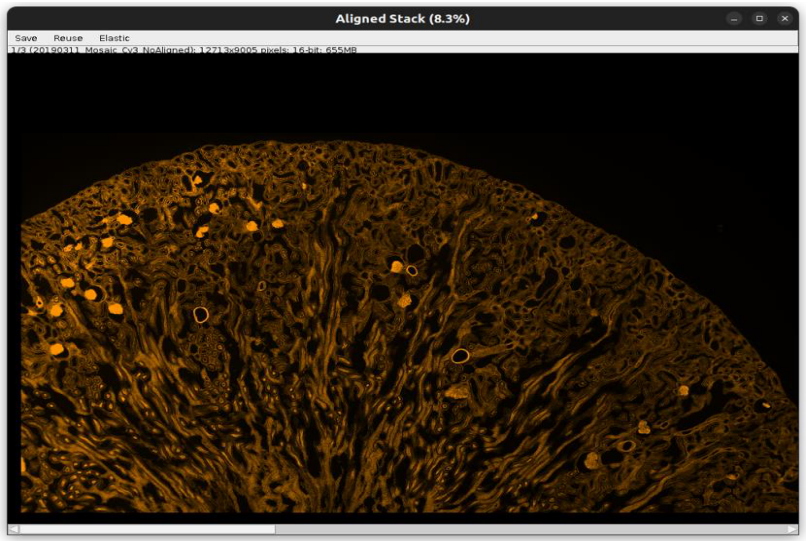



*Figura 1.19: Finestra di output.*

Nel caso in cui volessimo vedere le nostre immagini sovrapposte, dopo averle allineate, ci basterà adoperate *ImageJ/Fiji,* così facendo potremo ottenere all’interno della stesso stack le varie immagini allineate sovrapposte (Figura 1.20).

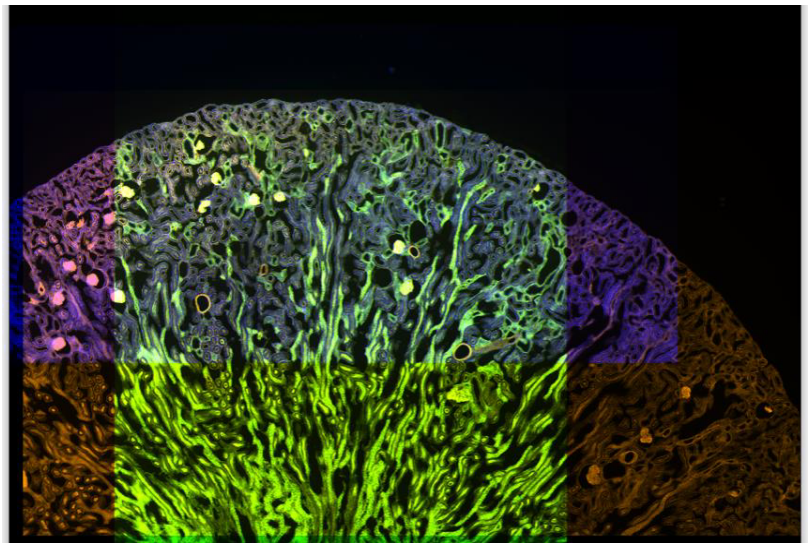


*Figura 1.20: Finestra di output, con immagini allineate sovrapposte.*

### 1.5.15 Salvataggio delle immagini

Una volta che abbiamo ottenuto il nostro stack allineato, possiamo effettuare una serie di operazioni, tra cui il *salvataggio* delle immagini, per fare ciò basterà semplicemente cliccare sulla voce "Save" nel menu bar e selezionare quali tra le immagini di output si vorranno salvare (Figura 1.21). Dopo aver scelto quali immagini salvare verrà chiesto in che directory salvare le immagini, in questo modo nel percorso specificato troveremo una cartella denominata "*YYYYMMDD_HHMM_DS4H_AlignedImages*".

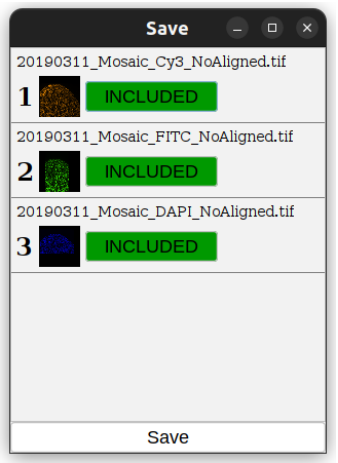



*Figura 1.21: Finestra per il salvataggio delle immagini.*

### 1.5.16 Riuso delle immagini

Dopo aver allineato le immagini, oltre al salvataggio, si può eseguire un'ulteriore operazione che consiste nell'utilizzare tutte o alcune delle immagini per ulteriori allineamenti utilizzando l'opzione "Reuse" nel menu bar. Una volta selezionata questa voce, verrà richiesto di selezionare le immagini che si desiderano utilizzare per questa operazione (Figura 1.22).

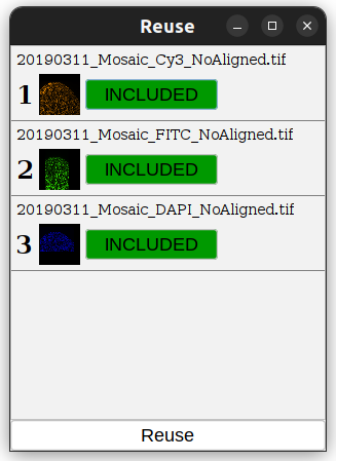



*Figura 1.22: Finestra per il riuso delle immagini.*

### 1.5.17 Applicazione algoritmo di deformazione elastica

Dopo aver allineato le immagini, si può eseguire l'ultima operazione che consiste nell'applicare l'algoritmo di deformazione elastica *bUnwarpJ*. Per farlo, si deve semplicemente cliccare sull'opzione "*Elastic*" nel menu. In automatico, il nostro programma utilizzerà le impostazioni configurate nella finestra principale come parametri di input per l'algoritmo.

### 1.5.18 Allineamento automatico

Dopo aver caricato correttamente le immagini nel nostro plugin, si può facilmente utilizzare la funzionalità di allineamento automatico cliccando sul pulsante "Automatic Alignment". Verrà visualizzata immediatamente una finestra di caricamento che mostrerà lo stato di avanzamento dell'allineamento (Figura 1.18). Una volta completato con successo l'allineamento, verrà mostrata la *finestra di output* (Figura 1.19).

# 2 Algoritmi per registrazione automatica

In questo capitolo, sarà presentato il primo modulo di sviluppo della mia Tesi, ovvero il modulo implementato per l'allineamento automatico delle immagini. La funzionalità è stata realizzata utilizzando la rinomata libreria di Computer Vision chiamata "*OpenCV*". Nel corso del capitolo, verranno fornite dettagliate spiegazioni sulle scelte implementative effettuate, che sono fondamentali per la costruzione di questa funzionalità.

## 2.1 Computer Vision

La Computer Vision è una disciplina all'interno del campo dell'Intelligenza Artificiale (AI) che permette alle AI di prendere decisioni basate su ciò che "vedono". L'obiettivo principale della Computer Vision è quello di imitare il nostro senso della vista. Analogamente a noi esseri umani, che fin dalla nostra infanzia abbiamo imparato a classificare e distinguere informazioni attraverso retine, nervi ottici e la corteccia visiva, la Computer Vision utilizza algoritmi per raggiungere lo stesso scopo [19].

## 2.2 OpenCV Extra Features Library

*OpenCV* include numerosi moduli, tutti utilizzabili senza alcuna licenza, come gli algoritmi: *AKAZE*, *KAZE*, *BRISK* e *ORB*, mentre altri algoritmi sono soggetti ad alcune ad eccezione. Queste eccezioni sono racchiuse nella libreria chiamata "*OpenCV Contrib"*, che contiene moduli con funzionalità che potrebbero non essere stabili o che richiedono una licenza a pagamento per l'uso commerciale, ma non per prodotti successivamente resi open-source. All'interno di questa libreria sono presenti due degli algoritmi utilizzati in questo lavoro di Tesi per creare il modulo di allineamento automatico delle immagini: *SURF* e *SIFT*.

## 2.3 Compilazione della sorgente

*OpenCV* è disponibile in forma precompilata per Android, iOS e Windows, ma per qualsiasi altra piattaforma (o quando si desidera utilizzare la libreria *OpenCV Contrib*) è necessario compilare il codice sorgente personalmente. Per implementare il modulo di allineamento automatico delle immagini, ho dovuto scaricare entrambi i sorgenti e generare un file JAR e le relative librerie per le

piattaforme di esecuzione, ad esempio i file *.dll per Windows, i file *.so per Linux e i file *.dylib per MacOS X con architettura ARM. Per questo processo, è necessario avere una distribuzione di Java installata nel sistema operativo, scaricare il programma "CMake" e utilizzare un terminale. Dopo aver scaricato le librerie OpenCV e OpenCV Contrib in formato zip e averle estratte, è utile mantenere entrambe le cartelle risultanti all'interno di una cartella padre per comodità. Successivamente, è necessario aprire il terminale e impostare la cartella padre come percorso corrente. A questo punto, è possibile eseguire sequenzialmente i comandi specificati come Codice 2.1, Codice 2.2 e Codice 2.3. I file prodotti come risultato di questi comandi saranno posizionati nella cartella "Nome Cartella Padre/build/lib" e potranno essere utilizzati nel proprio progetto. Il nostro plugin include già le librerie OpenCV precompilate e le caricherà automaticamente in una cartella temporanea del sistema operativo. In questo modo, sarà possibile utilizzare OpenCV senza incontrare problemi, poiché la libreria specifica per il sistema operativo corrente verrà gestita in modo automatico [20].

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

*Codice 2.1: Creazione cartella build.*

```
cmake -DCMAKE_SYSTEM_PROCESSOR=x86 \
-DCMAKE_OSX_ARCHITECTURES=x86 \
-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/x86/libawt.so \
-D JAVA_JVM_LIBRARY=$JAVA_HOME/jre/lib/x86/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 2.2: Creazione della build su architettura x86.*

```
make -j8
```

*Codice 2.3: Esempio compilazione sorgente per piattaforma x86.*

## 2.4 OpenCV Loader

*DS4H Image Alignment* è un plugin progettato per essere multipiattaforma, e questo requisito deve essere rispettato anche per le librerie di terze parti importate, come *OpenCV*. Per gestire questa proprietà del nostro plugin, ho creato una classe chiamata "OpenCVLoader" nel pacchetto "opencvManager/opencvLoader". Questa classe viene utilizzata all'interno di un controller specifico, in quanto è stato adottato il pattern architetturale **MVC** (Model-View-Controller). In particolare, la classe presenta un metodo chiamato "loadOpenCV" (Codice 2.4), che viene invocato prima che il plugin sia lanciato. Ciò garantisce che la libreria compilata sia caricata correttamente nella cartella temporanea del sistema operativo corrente. Questo metodo tiene conto del sistema operativo in uso, poiché ogni sistema operativo ha il proprio formato per le librerie.

```
public static void loadOpenCV(){
    if(OS.contains(WINDOWS)){
        OpenCVLoader.loadWindows();
    }else if(OS.contains(MAC)){
        final String arch = System.getProperty("os.arch").toLowerCase();
        if (arch.equals(ARM) || arch.startsWith(ARMV)) {
            OpenCVLoader.loadMacArm();
        } else {
            OpenCVLoader.loadMacIntel();
        }

    }else if(OS.contains(LINUX)){
        OpenCVLoader.loadLinux();
    }
}
```

*Codice 2.4: Implementazione del metodo OpenCVLoader.loadOpenCV().*

## 2.5 Algoritmi di allineamento automatico

Nel nostro software, abbiamo incluso diverse opzioni di allineamento automatico per consentire la scelta di algoritmi specifici in base alle immagini caricate. Questa scelta è stata motivata dal fatto che alcuni algoritmi funzionano meglio in determinate condizioni di illuminazione o con tipologie specifiche di immagini. Spiegheremo in seguito la strategia adottata per gestire questa varietà di modelli e come il processo di allineamento funziona nel frattempo spieghiamo l'idea che sta alla base di questo procedimento. Nel nostro algoritmo specifico, abbiamo adottato un approccio di confronto tra le caratteristiche di due immagini:

- la prima immagine (denominata "Target") rimane invariata;
- La seconda immagine viene deformata in base alle proporzioni della prima.

Il metodo di allineamento automatico utilizzato si basa sull'impiego degli algoritmi *SIFT* e *SURF*, che sono implementati nella libreria *OpenCV*. Questi algoritmi sono impiegati per rilevare e confrontare le caratteristiche di due immagini. Successivamente, le immagini vengono sottoposte a una fase di pre-elaborazione in cui vengono aggiunti pixel di sfondo in tutte le direzioni, in questa maniera evitiamo la perdita di parti dell'immagine. Infine, viene applicato un algoritmo di allineamento manuale, che può essere di tipo traslativo, affine o proiettivo, a seconda delle necessità e a seconda di ciò che è stato scelto dalle impostazioni.

### 2.5.1 Codice in dettaglio

Prendendo come punto di partenza un diagramma di flusso, procederemo a descrivere in modo approfondito le principali sezioni di codice relative al modulo sviluppato per l'allineamento automatico (Figura 2.1).


*Figura 2.1: Diagramma di flusso del modulo di allineamento automatico.*

### 2.5.2 Evento allineamento automatico

Prima di avviare l'allineamento automatico, è necessario caricare tutte le immagini su cui si desidera lavorare all'interno del plugin. Nel caso in cui l'utente non carichi alcuna immagine, il bottone per l'allineamento rimarrà disabilitato fino a quando non saranno caricate almeno due immagini (Figura 1.1). Il processo di allineamento inizia quando l'utente fa clic sul bottone nella finestra principale. Una volta premuto il bottone, viene immediatamente eseguito il suo "Handler", il cui scopo è raccogliere tutte le informazioni necessarie per eseguire l'allineamento. Queste informazioni sono prese dalla finestra delle impostazioni dell'allineamento automatico. Dopo aver ottenuto tutte le informazioni necessarie, il controller per l'allineamento aprirà un nuovo thread per gestire l'intero processo di allineamento automatico, consentendo comunque l'interazione con altre finestre. La prima operazione di allineamento consiste nell'individuazione dei punti chiave su ciascuna immagine tramite l'utilizzo del *PointDetector* selezionato, di default è *SURF*. Successivamente, viene eseguita la fase di pre-elaborazione e infine viene applicata la trasformazione finale delle immagini ("warping"). Una volta completato l'intero processo, viene mostrata all'utente la finestra di output.

### 2.5.3 Individuazione dei KeyPoint

Nel processo di sviluppo degli algoritmi di ricerca dei *keypoint* per ogni immagine, ho scelto di adottare il pattern **"***Template Method*". In questo pattern, si ha una classe base (la classe padre) che definisce uno scheletro o un algoritmo generale per un'operazione, lasciando alcune parti di essa da implementare alle classi figlie. Questo pattern è stato particolarmente adatto per gestire le operazioni comuni a tutti gli algoritmi, come i metodi per ridimensionare l'immagine e altre operazioni di pre-elaborazione. Nella mia implementazione, ho creato una classe base che contiene un metodo *astratto* per l'algoritmo di ricerca dei *keypoint*, il metodo è chiamato "***detectPoint***" (Figura 2.2). Successivamente, ho creato due classi figlie che estendono la classe base e forniscono l'implementazione specifica per il metodo di ricerca dei punti chiave in ogni singola immagine, una classe per ogni singolo Detector. Queste classi figlie sono specializzate per diversi algoritmi di ricerca dei *keypoint* e implementano i dettagli specifici per ciascun algoritmo. Utilizzando questo pattern, ho potuto gestire efficacemente le parti comuni dell'algoritmo e allo stesso tempo fornire la flessibilità necessaria per personalizzare l'implementazione per ogni algoritmo di ricerca dei *keypoint*. Inoltre, l'uso di un'interfaccia comune ha semplificato l'integrazione e la manutenzione del codice, permettendomi di estendere facilmente il sistema con nuovi algoritmi di ricerca dei *keypoint* in futuro. Di seguito andrò a descrivere il funzionamento dei due detector utilizzati: *SIFT* e *SURF*.

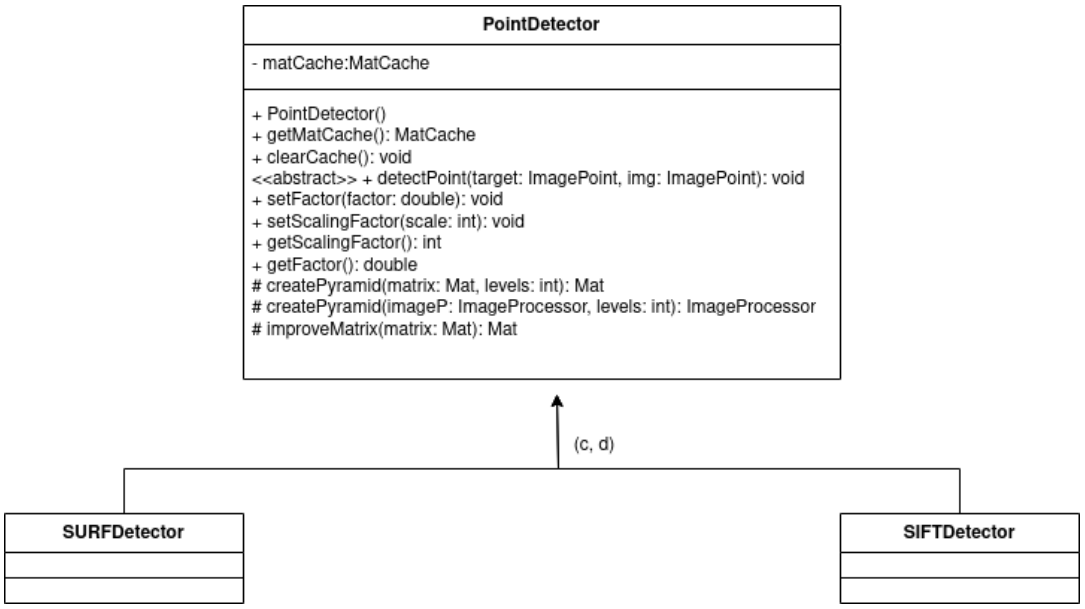



*Figura 2.2: Struttura dei Detector.*

### 2.5.4 SIFT Scale Invariant Feature Transform

Nel 1999, D.G. Lowe ha sviluppato l'algoritmo chiamato "*Scale Invariant Feature Transform*" (*SIFT*) [21]. Esso si basa sull'algoritmo denominato "*Differenze di Gaussiane*" (*DoG*) per individuare corrispondenze tra immagini. Le *DoG* vengono utilizzate per individuare punti d'interesse che sono invarianti rispetto alla scala e all'orientamento. Per ogni gruppo di punti, viene cercato un modello adeguato per determinarne la posizione e la scala, definendo così dei punti chiave basati sulla stabilità delle misurazioni effettuate. Ad ogni punto chiave viene assegnato un orientamento, basato sulla direzione del gradiente locale dell'immagine. I gradienti locali dell'immagine vengono misurati ad una scala specifica e vengono trasformati in una rappresentazione che tiene conto dei cambiamenti di illuminazione e delle distorsioni della forma. La rappresentazione spazio-scala di un'immagine è definita da una funzione $L\ (x,\ y,\ \sigma)$ che è prodotta dalla convoluzione di una variabile spazio-scala Gaussiana $G$ con un'immagine in input $I$ [21].

$$L(x, y, \sigma) = G(x, y, \sigma) * I(x, y)$$

dove la funzione $G$ è definita da:

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

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

$$\begin{aligned} D(x, y, \sigma) &= (G(x, y, k\sigma) - G(x, y, \sigma)) * I(x, y) \\ &= L(x, y, k\sigma) - L(x, y, \sigma) \end{aligned}$$

### 2.5.5 SURF Speed Up Robust Feature

L'algoritmo "*SURF*" (*Speed Up Robust Feature*) è nato nel 2006 [22]. 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 dato da $I_\Sigma$ ad una posizione $x = (x, y)^T$, ed è la somma di tutti i pixel presenti all'interno di una porzione rettangolare di un'immagine presa in input:

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

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

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

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

### 2.5.6 Pre-Elaborazione delle immagini

Durante le analisi istologiche, ogni frammento di tessuto racchiude preziose informazioni. Anche se può sembrare insignificante, la perdita di una piccola sezione di immagine costituisce un grave danno per gli operatori di questo settore. Tale problematica può portare a una significativa perdita di dati e informazioni, causando errori e danni irreparabili. La fase di pre-elaborazione delle immagini risolve il problema di perdere parti significative durante la fase finale dell'allineamento, a causa delle dimensioni diverse delle immagini. Per eseguire questa operazione, è necessario aver identificato i keypoint su tutte le immagini per l'allineamento. Durante questo processo, solo un'immagine, l'immagine target, viene espansa. In questo modo, quando si esegue la trasformazione geometrica su tutte le immagini, viene specificata la dimensione finale desiderata, corrispondente alla dimensione del target. Ciò garantisce che tutte le immagini allineate abbiano le stesse dimensioni e che su ogni immagine non vadano perse informazioni.

### 2.5.7 Espansione immagine

Il processo con cui viene eseguita questa operazione è molto complesso. L'immagine target è l'immagine soggetta all'espansione. Utilizzando specifiche funzioni della libreria *OpenCV*, è possibile determinare di quanti pixel l'immagine target verrà espansa in tutte le sue direzioni rispetto ad ogni immagine (Figura 2.3). Ciò consente di creare una nuova immagine target di dimensioni maggiori, contenente nei nuovi bordi aggiunti dei semplici pixel di colore nero. Tuttavia, l'espansione dell'immagine influisce sui keypoint individuati nella fase precedente. Questo problema deriva dal fatto che ogni keypoint dipende dalla dimensione dell'immagine target utilizzata nel processo precedente per determinare quali fossero i keypoint migliori individuati tramite gli algoritmi automatici. Per sopperire a questo problema si è pensato di scalare tutti i punti individuati di ogni singola immagine ogni volta che l'immagine target subisce un'espansione, così facendo tutti i punti vengo posizionati in delle nuove coordinate permettendo poi di eseguire il "warping" senza perdita di dati.

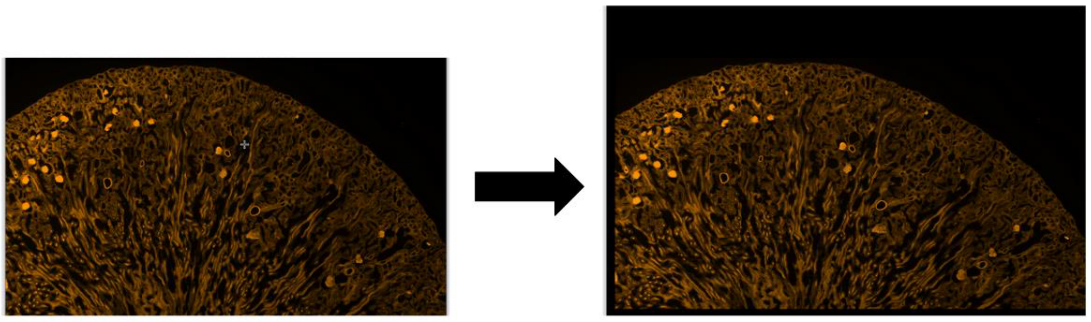


Immagine originale

Immagine con bordi estesi

*Figura 2.3: Esempio di estensione di bordi.*

### 2.5.8 Warping delle immagini

La fase finale dell'allineamento prevede di effettuare una trasformazione su ogni immagine impiegata per l'allineamento eccetto l'immagine target. All'interno di questa fase viene effettuata una trasformazione geometrica a seconda del tipo di algoritmo di allineamento selezionato nel menù delle impostazioni per l'allineamento automatico (Figura 1.5). Grazie alla fase precedentemente descritta è possibile quindi mantenere tutti i dati presenti in ogni immagine senza perdere parti di immagine potenzialmente significativa, inoltre *DS4H Image Alignment* è in grado di allineare tutte le immagini mantenendo le loro proprietà iniziali come profondità dei pixel, LUT e altre proprietà, a differenza dei *tool competitors*.

## 2.6 Gestione Big Data

Un problema significativo associato all'utilizzo dei microscopi è la gestione dell'allineamento delle immagini di grandi dimensioni, che possono superare anche i cento megabyte. Gli algoritmi utilizzati per individuare automaticamente i punti chiave richiedono notevoli risorse computazionali, anche per immagini di pochi kilobyte. Per affrontare questa sfida, ho adottato una strategia basata sulla tecnica ben nota della "*scaling*" all'interno della *Computer Vision*. La tecnica dello "*scaling*" consente di modificare le dimensioni di un'immagine al fine di ridurre il consumo di memoria e utilizzare algoritmi complessi senza compromettere l'esecuzione del programma. In particolare, ho impiegato una tecnica piramidale. L'obiettivo principale della tecnica piramidale è ridurre efficientemente le dimensioni di un'immagine, preservando le sue caratteristiche salienti. Ciò permette di individuare i punti chiave fondamentali senza rischiare di perderli. La tecnica piramidale prevede la creazione di una serie di immagini ridotte, chiamate livelli o scale, in cui ogni

livello successivo ha una risoluzione inferiore rispetto al precedente. Di solito, si parte dall'immagine originale come livello base (il livello più basso della piramide) e si ottengono i livelli successivi riducendo progressivamente le dimensioni dell'immagine (Figura 2.4 [23]).

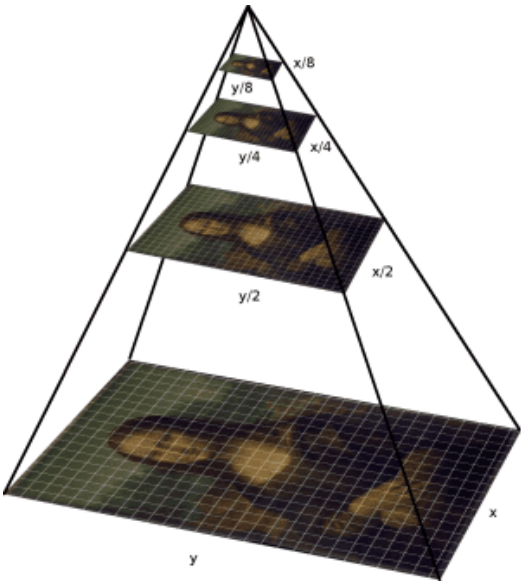



*Figura 2.4: Esempio di scaling piramidale.*

Il livello di scaling può essere gestito attraverso il menu degli algoritmi automatici (Figura 1.5). Come valore predefinito è impostato 4, poiché ho osservato che la dimensione media delle immagini su cui si lavora è nell'ordine delle centinaia di megabyte. La funzione di scaling è stata implementata all'interno della classe "*PointDetector*" (Codice 2.5). In questo modo, ogni classe figlia può chiamare questa funzione prima di eseguire il metodo "*detectPoint"* al fine di ridimensionare le immagini e sfruttare a pieno i più complessi algoritmi per l'individuazione di keypoint.

```
protected Mat createPyramid(final Mat matrix, final int levels){
    Size lastSize = matrix.size();
    IJ.log("[POINT DETECTOR] Resize original matrix by: " + levels + " times.");
    for(int i = 1; i < levels; i++) {
        Imgproc.resize(matrix, matrix,
                new Size(lastSize.width / 2, lastSize.height / 2),
                Imgproc.INTER_AREA);
        lastSize = matrix.size();
    }
    IJ.log("[POINT DETECTOR] Matrix:" + matrix + ".");
    IJ.log("[POINT DETECTOR] Resize done.");
    return matrix;
}
```

*Codice 2.5: Codice per lo scaling piramidale implementato nella classe PointDetector.*

# 3 Algoritmi per la deformazione elastica

In questo capitolo, verrà presentato il secondo modulo sviluppato per la mia Tesi, che si occupa di eseguire deformazioni elastiche. Questa funzionalità è stata implementata utilizzando la libreria *bUnwarpJ*. Nel corso del capitolo, saranno fornite spiegazioni dettagliate sulla ragione dietro l'utilizzo di questo algoritmo e su come è stato integrato nel nostro software.

## 3.1 Problema delle deformazioni

Durante le analisi istologiche, l'utilizzo di diversi coloranti richiede l'esecuzione di numerosi risciacqui dei tessuti al fine di applicare coloranti diversi. Tuttavia, questa procedura provoca un significativo danneggiamento dei tessuti. I risciacqui ripetuti possono causare deformazioni o distanziamenti delle sezioni tissutali, rendendo difficile l'identificazione accurata delle diverse strutture presenti nel campione sotto osservazione, come la separazione dei diversi strati cellulari o l'individuazione di confini precisi tra tessuti adiacenti all'interno dello stesso soggetto. Questa situazione può compromettere l'analisi morfologica e la caratterizzazione delle alterazioni tissutali. Al fine di affrontare questo problema, si è deciso di adottare l'approccio di utilizzare la libreria *bUnwarpJ*.

### 3.1.1 bUnwarpJ

*bUnwarpJ* è un algoritmo avanzato per la registrazione elastica delle immagini, sviluppato come un plugin per *ImageJ*, un software di elaborazione delle immagini. Il suo obiettivo principale è quello di allineare in modo elastico due immagini, denominate A e B. Il processo di registrazione inizia con l'immagine A, che viene deformata elasticamente per renderla simile all'immagine B. Questo viene ottenuto applicando trasformazioni locali che consentono all'immagine A di adattarsi meglio all'immagine B, tenendo conto delle differenze locali di forma e intensità tra le due immagini. *bUnwarpJ* utilizza un approccio di "deformazione di griglia" per calcolare la deformazione locale dell'immagine A rispetto all'immagine B. Questo viene realizzato dividendo l'immagine in una griglia di celle e calcolando le trasformazioni locali di ogni cella per allineare i punti di controllo. Le deformazioni locali vengono quindi propagate alle regioni circostanti, consentendo una deformazione fluida e coerente dell'intera immagine. Durante il processo di registrazione, bUnwarpJ tiene conto anche delle differenze di intensità tra le immagini A e B. Ciò consente di

effettuare un allineamento accurato non solo delle strutture geometriche, ma anche delle caratteristiche di intensità, come i gradienti di colore o le differenze di intensità locali [24]. Un'operazione preliminare che deve essere eseguita prima di impiegare questa libreria è l'allineamento delle immagini. Se si volesse utilizzare *bUnwarpJ* semplicemente come plugin di *ImageJ/Fiji*, risulterebbe molto lungo il processo in quanto tramite il plugin è possibile effettuare la deformazione elastica con sole due immagini alla volta. Tramite il nostro software invece, in automatico viene richiamato *bUnwarpJ* su tutte le immagini allineate.

### 3.1.2 Integrazione di bUnwarpJ

La modalità con la quale è stata integrata questa libreria è stata pensata in modo da rendere l'esperienza utente il più semplice possibile. Qui di seguito viene mostrato il flusso con la quale la libreria viene impiegata (Figura 3.1).

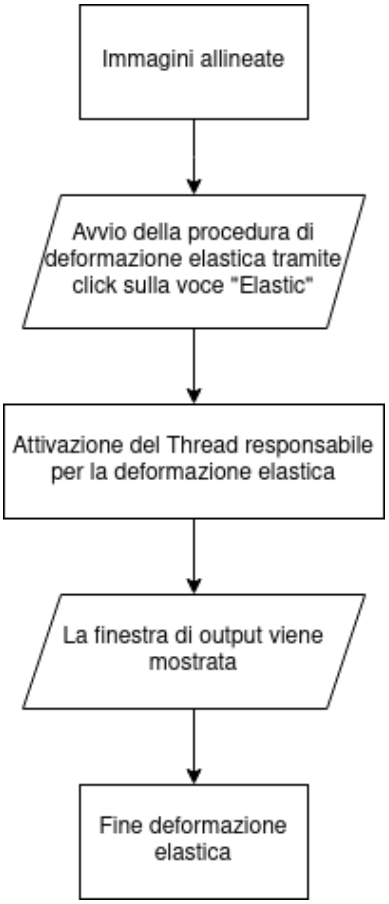



*Figura 3.1: Diagramma di flusso per la deformazione elastica.*

La prima operazione da eseguire all'interno del plugin, dopo aver caricato le immagini (Figura 3.2), per utilizzare la libreria *bUnwarpJ* è l'allineamento delle immagini secondo la modalità preferita. Una volta che le immagini sono state correttamente allineate, è possibile utilizzare l'algoritmo di

deformazione elastica selezionando l'opzione "*Elastic*" nel menu della barra degli strumenti nella finestra di output (Figura 3.3).


*Figura 3.2: Immagini da allineare.*

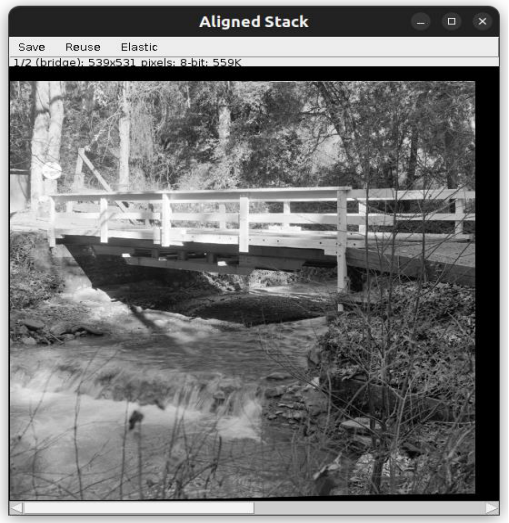



*Figura 3.3: Finestra di output dopo aver allineato le immagini.*

Dopo aver selezionato l'opzione "*Elastic*", verrà mostrata all'utente la finestra di caricamento (Figura 1.18). Una volta completata l'esecuzione dell'algoritmo, verrà visualizzata una nuova finestra di output (Figura 3.4) contenente le immagini deformate risultanti (Figura 3.5).

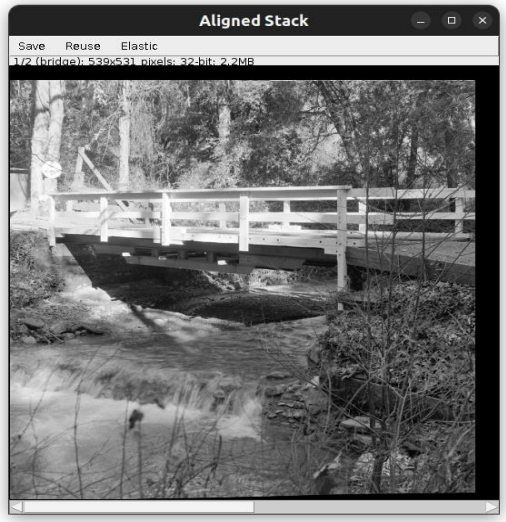



*Figura 3.4: Finestra di output dopo aver applicato bUnwarpJ.*


*Figura 3.5: Immagini dopo aver deformato tramite bUnwarpJ.*

Da come si può vedere l'immagine "deformed-bridge" non presenta più alcuna deformazione, questo grazie alla libreria *bUnwarpJ* ed alla semplicità con la quale è possibile utilizzarlo all'interno del nostro plugin.

# 4 Gestione ottimizzata di immagini multicanale

Nel presente capitolo, sarà presentato il terzo ed ultimo modulo sviluppato in questo lavoro di Tesi. Fornisce una spiegazione approfondita del processo di gestione delle immagini all'interno dell'applicativo. Durante il capitolo, saranno fornite spiegazioni dettagliate sul caricamento di ciascuna immagine nel programma e sulla loro gestione durante le trasformazioni geometriche subite durante gli allineamenti.

## 4.1 Immagini digitali

Il dato principale con cui il software opera è l'immagine digitale. Un'immagine digitale rappresenta numericamente un'immagine visiva ed è costituita da un insieme di dati che descrivono le caratteristiche visive dell'immagine, come colore, luminosità e disposizione dei pixel. I pixel sono gli elementi fondamentali che compongono l'immagine e contengono informazioni sul colore e la luminosità. Le immagini digitali possono essere a colori, in scala di grigi o in bianco e nero, a seconda del numero di bit assegnati a ciascun pixel per rappresentare il colore. Le immagini digitali possono essere salvate in diversi formati di file, come JPEG, PNG o TIFF, ognuno con caratteristiche specifiche e compressione dei dati.

## 4.2 Immagini monocanale

In un'immagine monocanale, ciascun pixel rappresenta l'intensità luminosa di un punto specifico dell'immagine. Questa intensità può variare da un valore minimo (nero) a un valore massimo (bianco), con sfumature di grigio tra questi due estremi. Ad esempio, un pixel con un valore di intensità basso apparirà più scuro, mentre un pixel con un valore di intensità elevato apparirà più chiaro. I valori che si possono trovare all'interno di un'immagine monocanale variano tra un minimo di 0 e un massimo di 255, dal momento in cui si ha un singolo byte per pixel (Figura 4.1).

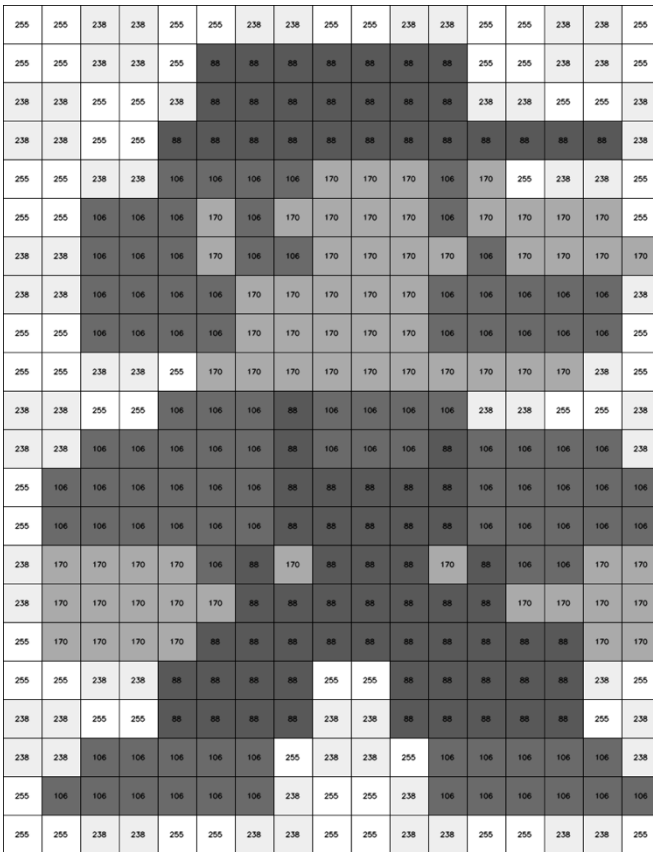



*Figura 4.1: Esempio immagine monocanale.*

## 4.3 Immagini multicanale

Un'immagine multicanale è un tipo di immagine digitale in cui ogni pixel è rappresentato da più di un valore numerico. In genere, si tratta di immagini a colori in cui vengono utilizzati più canali per rappresentare le diverse componenti del colore (Figura 4.2). Nelle immagini multicanale, ogni canale rappresenta una specifica informazione di colore. I canali più comuni sono i canali rosso (R), verde (G) e blu (B), utilizzati per creare immagini RGB. In un'immagine RGB, ogni pixel è rappresentato da tre valori numerici, uno per ogni canale, che indicano l'intensità dei rispettivi colori primari.

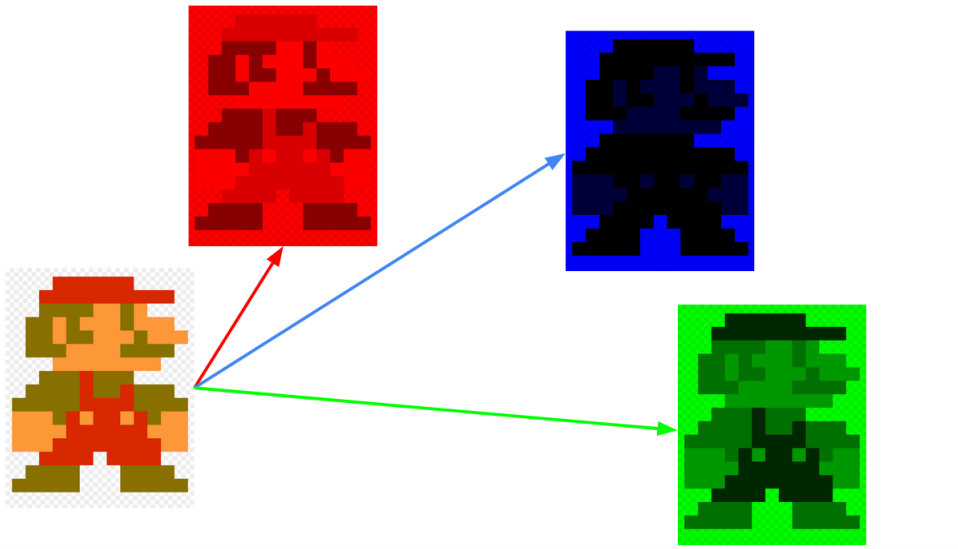


*Figura 4.2: Esempio immagine multicanale.*

## 4.4 Formati delle immagini

I formati delle immagini digitali servono a specificare la struttura e la codifica dei dati di un'immagine, consentendo la memorizzazione, la trasmissione e la visualizzazione corretta delle immagini su dispositivi e piattaforme diverse. Ogni formato ha le sue caratteristiche e vantaggi, adatti a diversi scopi e contesti di utilizzo. I formati più comuni con il quale il nostro software deve andare a lavorare sono:

- **PNG** (Portable Network Graphics), il formato PNG è ampiamente utilizzato per immagini raster con compressione senza perdita di qualità. Supporta la trasparenza, consentendo di rappresentare immagini con sfondi trasparenti.È ideale per immagini con bordi netti, testo e grafica lineare. È adatto per immagini web, grafica vettoriale e immagini con dettagli nitidi. Tuttavia, poiché il formato PNG non utilizza compressione con perdita di qualità, i file possono essere più grandi rispetto ad altri formati [25].
- **JPEG** (Joint Photographic Experts Group), il formato JPEG è ampiamente utilizzato per immagini raster con compressione con perdita di qualità. È ottimizzato per la compressione di immagini fotografiche, in particolare immagini a colori ad alta risoluzione. Supporta una vasta gamma di colori e consente di regolare il livello di compressione per bilanciare la qualità dell'immagine e la dimensione del file. Poiché utilizza la compressione con perdita di qualità, alcune informazioni vengono scartate e possono verificarsi artefatti visibili quando l'immagine viene compressa in modo significativo. È il formato più comune per le immagini web e le immagini con sfumature o transizioni di colore morbide [26].

- **TIFF** (Tagged Image File Format), Il formato TIFF è un formato flessibile e ampiamente utilizzato per immagini raster. Supporta la compressione senza perdita di qualità, consentendo di mantenere l'integrità dei dati dell'immagine. Può memorizzare immagini in scala di grigi, immagini a colori e immagini multicanale. È adatto per immagini ad alta profondità di bit, come immagini mediche o immagini che richiedono una precisione elevata. Il formato TIFF può avere dimensioni di file più grandi rispetto ad altri formati a causa della sua flessibilità e capacità di memorizzare dati dettagliati. È spesso utilizzato in applicazioni professionali, come la grafica, la fotografia, l'archiviazione di immagini mediche e la stampa [27].

## 4.5 Caricamento delle immagini

Come abbiamo appreso dai moduli precedenti, per utilizzare il programma è necessario caricare una serie di immagini. Durante la selezione dei file all'interno del nostro plugin, potrebbe accadere di selezionare erroneamente anche file che non sono immagini effettive, come ad esempio file PDF, file di testo o altri tipi di file. Una volta che l'utente ha selezionato l'insieme di file da caricare, viene eseguita una scansione dettagliata per verificare se ciascun file è effettivamente un'immagine. Questa verifica viene effettuata controllando l'estensione del file (Codice 4.1).

```
public static boolean checkImage(final File file){
    if(Objects.nonNull(file) && file.isFile()) {
        final String fileExtension = FilenameUtils.getExtension(file.getName());
        if(!fileExtension.isEmpty() && CheckImage.EXTENSIONS.contains(fileExtension.toLowerCase())){
            return CheckImage.checkSize(file);
        }
    }
    return false;
}
```

*Codice 4.1: Codice per il controllo di immagine.*

Se il file osservato non è un'immagine, viene semplicemente ignorato e non viene caricato nel programma. Tuttavia, se il file è un'immagine effettiva, viene utilizzato come base per inizializzare la classe "*ImagePoint*". Questa classe viene successivamente utilizzata per gestire ogni singola immagine all'interno del programma (Codice 4.2). In questo modo, solo i file che rappresentano immagini valide vengono elaborati e gestiti dal programma, garantendo un funzionamento ottimale.

```
public static List<ImagePoints> fromPath(final List<File> paths){
    return paths.parallelStream()
            .filter(File::isFile)
            .filter(CheckImage::checkImage)
            .flatMap(file -> {
                try{
                    return ImagingConversion.isMulti(file);
                }catch (final IOException exception){
                    return Stream.empty();
                }
            })
            .map(File::getPath)
            .map(ImagePoints::new)
            .collect(Collectors.toList());
}
```

*Codice 4.2: Codice per caricare solamente le immagini nel plugin.*

## 4.6 Gestione multi-TIFF

Se l'immagine selezionata è un TIFF multi-pagina [28], si tratta di un formato di file immagine che consente di archiviare più pagine o immagini all'interno di un singolo file TIFF. Questo formato offre vantaggi in termini di organizzazione e gestione delle immagini correlate. In un file TIFF multi-pagina, ogni pagina o immagine è rappresentata come un'entità separata all'interno del file. Ogni pagina può contenere un'immagine a colori, in scala di grigi o in bianco e nero, con diverse profondità di colore e formati. Questo permette di archiviare una varietà di tipologie di immagini all'interno dello stesso file. Tuttavia, il problema con il TIFF multi-pagina è che non esiste un formato TIFF specifico per indicare la presenza di più immagini all'interno del file. È necessario fare uso di librerie specializzate per determinare se il file contiene o meno più immagini. Per affrontare questa situazione, nel programma è stata implementata una funzione dedicata che controlla la presenza di una molteplicità di immagini nel caso in cui il formato dell'immagine sia TIFF. Questo controllo e la gestione delle immagini all'interno del file sono stati realizzati utilizzando la libreria "TIFFUtilities" (Codice 4.3). In questo modo, il programma è in grado di riconoscere e gestire correttamente i file TIFF multi-pagina, consentendo una corretta manipolazione delle immagini contenute all'interno del file. L'utilizzo della libreria "TIFFUtilities" offre le funzionalità necessarie per l'estrazione e la manipolazione delle singole pagine o immagini all'interno del file TIFF, per ogni immagine estratta si inizializza una classe “ImagePoint” diversa, in questo modo si potrà lavorare su tutte le immagini.

```
private static Stream<File> isMulti(final File file) throws IOException {
    if (CheckImage.isTiff(file) && TIFFUtilities.getPages(ImageIO.createImageInputStream(file)).size() != 1) {
        final String dir = DirectoryCreator.createTemporaryDirectory(ImagingConversion.TMP_DIRECTORY_NAME);
        final List<File> files = ImagingConversion.split(file, new File( ImagingConversion.TMP_DIRECTORY+ "/" + dir),
                FilenameUtils.removeExtension(file.getName()));
        return files.stream();
    } else {
        return Stream.of(file);
    }
}
```

*Codice 4.3: Codice per controllare se si tratta di una TIFF multi-pagina.*

### 4.6.1 Gestione delle immagini nel plugin

Per poter utilizzare appieno la libreria di Computer Vision *OpenCV*, è fondamentale lavorare con la classe *Mat* [29]. Senza l'utilizzo di questa classe, sarebbe impossibile sfruttare qualsiasi funzione offerta da *OpenCV*. La classe *Mat* è utilizzata per rappresentare e gestire le immagini. Ogni elemento della matrice corrisponde a un singolo pixel dell'immagine. La profondità di ogni pixel varia in base al tipo di immagine utilizzata. Le profondità che possono essere utilizzate sono:

- 8 bit, rappresentato dal tipo byte in Java;
- 16 bit, rappresentato dal tipo short in Java;
- 32 bit, rappresentato dal tipo float in Java;
- 24 bit, viene utilizzato per rappresentare le immagini RGB, si utilizza il tipo byte Java per ogni canale.

All'interno del software *ImageJ/Fiji*, le immagini sono rappresentate utilizzando una classe denominata *ImageProcessor* [30]. L'ImageProcessor in *ImageJ/Fiji* è un insieme di strumenti e funzioni che consentono di manipolare ed elaborare le immagini digitali. Questa classe offre una vasta gamma di funzionalità per il trattamento delle immagini, come filtri, operazioni di segmentazione, misurazioni, correzioni di luminosità e contrasto, fusione di immagini, operazioni logiche, trasformazioni geometriche e molto altro. Tra le molte operazioni possibili, una fondamentale è l'ottenimento dell'insieme di tutti i pixel dell'immagine, specificando la profondità con cui ogni pixel viene memorizzato. Inoltre, la classe ImageProcessor definisce anche quattro classi figlie:

1) *ByteProcessor*, per rappresentare le immagini con profondità di 8 bit;
2) *ShortProcessor*, per rappresentare le immagini con profondità di 16 bit;
3) *FloatProcessor*, per rappresentare le immagini con profondità di 32 bit;
4) *ColorProcessor*, per rappresentare le immagini RGB con profondità di 24 bit;

Utilizzando le funzioni di *ImageProcessor* e i metodi forniti dalla classe Mat di *OpenCV*, è possibile effettuare la conversione tra le due classi mediante diverse funzioni. Ho ideato l'implementazione di due classi specifiche per gestire questa conversione di classi. La prima è *ImageProcessorMatConverter* (Codice 4.5), che permette di passare da un oggetto di tipo *ImageProcessor* a un oggetto di tipo Mat. La seconda classe è *MatImageProcessorConverter* (Codice 4.6), che invece consente di effettuare la conversione da un oggetto di tipo Mat a un oggetto di tipo *ImageProcessor*.

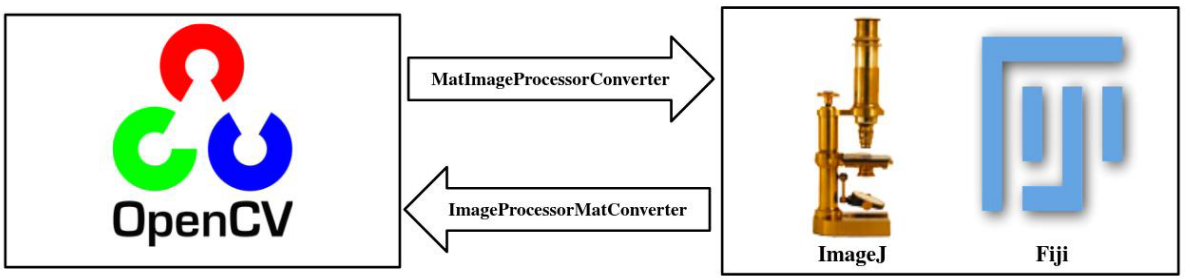
```
private static Mat toMat(final ShortProcessor sp){
    IJ.log("[TO MAT - SHORTPROCESSOR] From ShortProcessor");
    final Mat matrix = new Mat(sp.getHeight(), sp.getWidth(), CvType.CV_16U);
    IJ.log("[TO MAT - SHORTPROCESSOR] Matrix: " + matrix);
    matrix.put(0,0, (short[]) sp.getPixels());
    IJ.log("[TO MAT - SHORTPROCESSOR] Matrix created");
    return  matrix;
}
```

*Codice 4.4: Creazione della classe Mat da uno ShortProcessor.*

```
private static ByteProcessor makeByteProcessor(Mat matrix, final int width, final int height){
    IJ.log("[MAKE BYTEPROCESSOR] Creating ByteProcessor");
    final ByteProcessor ip = new ByteProcessor(width, height);
    if(matrix.channels() > 1) {
        Imgproc.cvtColor(matrix, matrix, Imgproc.COLOR_BGR2GRAY);
    }
    IJ.log("[MAKE BYTEPROCESSOR] Matrix: " + matrix);
    matrix.get(0,0, (byte[])ip.getPixels());
    matrix.release();
    matrix = null;
    IJ.log("[MAKE BYTEPROCESSOR] Finish creation ByteProcessor");
    return ip;
}
```

*Codice 4.5: Creazione di un ByteProcessor da una Mat.*

Tramite questo parallelismo tra *OpenCV* e *ImageJ/Fiji* (Figura 4.3) è stato possibile realizzare un plugin che permette di allineare tutte le immagini caricate mantenendo le loro proprietà, come profondità di ogni pixel, LUT, dimensione dei pixel ed altre informazioni che con altri tool competitor andrebbero perse. Questo rende il plugin *DS4H Image Alignment* un'ottima alternativa ai tool più famosi.




*Codice 4.3: Parallelismo tra OpenCV [31] e ImageJ [32] /Fiji [33] tramite le classi implementate.*

# 5 Risultati sperimentali

Il plugin sviluppato è stato validato attraverso dataset di immagini 2D reali acquisite con un microscopio confocale. In particolare, sono stati eseguiti una sequenza di test confrontando i risultati ottenuti usando l'algoritmo *SIFT*. I test sono stati condotti per valutare le prestazioni del software in diverse situazioni. Durante queste prove, è stata creata una tabella per riportare i dati ottenuti e le analisi condotte con le diverse metriche di valutazione. La tabella presenta una colonna per ogni dataset utilizzato. A sua volta ogni colonna è suddivisa in diverse sotto colonne che fanno riferimento ad ogni singola immagine utilizzata.

Questo dataset comprende:

1) XY1: immagine con una sovrapposizione approssimativa dello 0% rispetto alla principale;
2) XY2: immagine con una sovrapposizione approssimativa del 25% rispetto alla principale
3) XY3: immagine con una sovrapposizione approssimativa del 50% rispetto alla principale;
4) XY4: immagine con una sovrapposizione approssimativa del 75% rispetto alla principale;
5) XY5: immagine con una sovrapposizione approssimativa del 100% rispetto alla principale.

Durante l'analisi dei dati, sono stati registrati i risultati degli output, controllando se l'allineamento sia o meno avvenuto visivamente con successo. Nel caso in cui le immagini siano allineate correttamente troviamo come valore Y (acronimo di "Yes"), nel caso in cui invece le immagini non siano allineate correttamente troviamo il valore N (acronimo di "No"). In Figura 5.1 sono mostrate alcune delle immagini dei due dataset utilizzati.

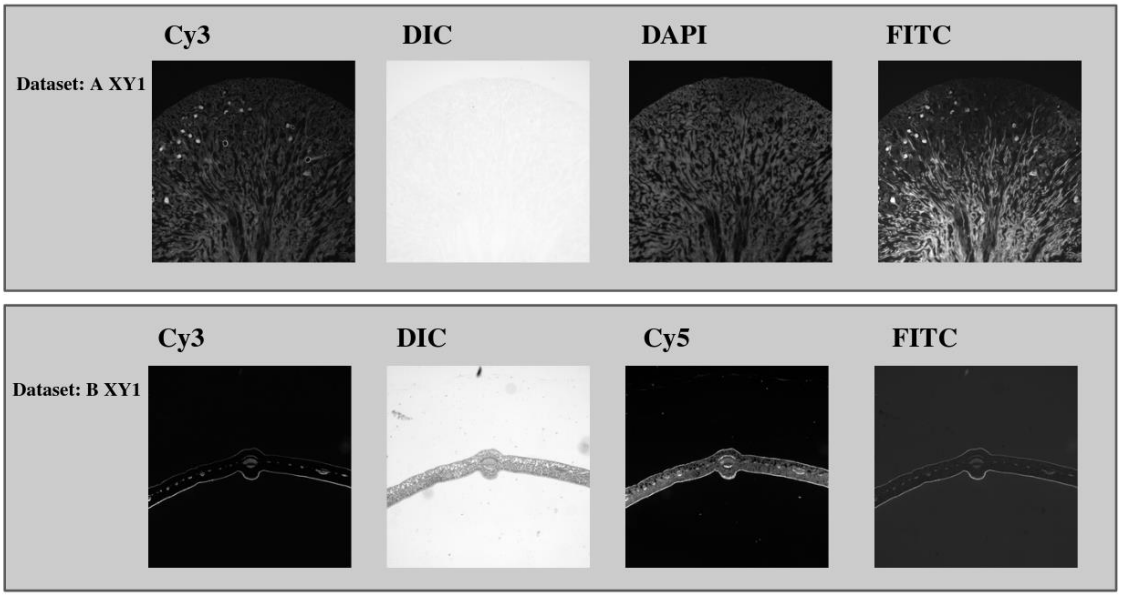



*Figura 5.1: Immagini rappresentative dei dataset utilizzati durante la prova sperimentale.*

## 5.1 Prima prova sperimentale

| | XY1 | | | | XY2 | | | | XY3 | | | | XY4 | | | | XY5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Cy3** | **DIC** | **DAPI** | **FITC** | **Cy3** | **DIC** | **DAPI** | **FITC** | **Cy3** | **DIC** | **DAPI** | **FITC** | **Cy3** | **DIC** | **DAPI** | **FITC** | **Cy3** | **DIC** | **DAPI** | **FITC** |
| **Cy3** | Y | N | Y | Y | Y | N | Y | Y | Y | N | Y | Y | Y | N | Y | Y | Y | N | Y | Y |
| **DIC** | N | Y | N | N | N | N | N | N | N | Y | N | N | N | N | N | N | N | N | N | N |
| **DAPI** | Y | N | Y | N | Y | N | Y | N | Y | N | Y | N | Y | N | Y | N | Y | N | Y | N |
| **FITC** | Y | N | Y | Y | Y | N | Y | Y | Y | N | Y | Y | Y | N | N | Y | Y | N | Y | Y |

*Tabella 5.1: Risultati allineamenti utilizzando il dataset A tramite SIFT.*

Nella prima prova sperimentale, abbiamo per prima cosa preso in considerazione le immagini del dataset A senza alcun tipo di spostamento, ossia con uno shift dello 0%. I risultati ottenuti evidenziano che utilizzando SIFT siamo stati in grado di allineare correttamente il 50% delle immagini, e questo risultato rimane constante persino quando gli spostamenti diventano significativi. Tuttavia, abbiamo osservato che si verifica una concentrazione di errori quando si tenta di allineare le immagini di tipo DIC. Questo può essere attribuito alle caratteristiche specifiche delle immagini DIC, che sono caratterizzate da un forte contrasto differenziale basato sull'interferenza della luce polarizzata [34]. Tale contrasto può influenzare la robustezza e la qualità delle caratteristiche identificate da SIFT. Poiché SIFT si basa sull'individuazione di punti chiave invarianti di scala [35], potrebbe non riuscire a rilevare in modo adeguato tali caratteristiche nelle immagini DIC a causa delle loro peculiarità di contrasto. Questo fenomeno spiega perché le immagini DIC non riescono ad essere allineate correttamente con il resto delle immagini del dataset.

## 5.2 Seconda prova sperimentale

| | XY1 | | | | XY2 | | | | XY3 | | | | XY4 | | | | XY5 | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | **Cy3** | **DIC** | **Cy5** | **FITC** | **Cy3** | **DIC** | **Cy5** | **FITC** | **Cy3** | **DIC** | **Cy5** | **FITC** | **Cy3** | **DIC** | **Cy5** | **FITC** | **Cy3** | **DIC** | **Cy5** | **FITC** |
| **Cy3** | Y | N | N | Y | Y | N | N | Y | Y | Y | N | Y | Y | N | Y | Y | Y | Y | Y | Y |
| **DIC** | N | Y | N | N | N | Y | N | N | N | N | N | N | N | N | N | N | N | N | N | N |
| **Cy5** | Y | N | Y | N | N | N | Y | N | N | Y | Y | N | N | N | Y | N | Y | N | N | Y |
| **FITC** | Y | N | N | Y | Y | N | Y | Y | Y | N | Y | Y | Y | Y | N | Y | Y | Y | Y | Y |

*Tabella 5.2: Risultati allineamenti utilizzando il dataset B tramite SIFT.*

Nella seconda prova sperimentale, abbiamo nuovamente per prima cosa considerato le immagini del dataset B senza alcuno spostamento. I risultati ottenuti dimostrano ancora una volta che l'utilizzo dell'algoritmo SIFT consente di allineare correttamente circa il 50% delle immagini. Tuttavia, è

interessante notare che in questo dataset la percentuale di immagini DIC allineate è leggermente più alta. Ciò suggerisce che la difficoltà nel rilevare i punti chiave nelle immagini DIC varia in modo più significativo a seconda della struttura specifica del tipo di immagine considerato. Come evidenziato nella figura del dataset B (Figura 5.1), è possibile distinguere chiaramente l'immagine dallo sfondo, nonostante sia di tipo *DIC*. Tutti i bordi sono facilmente individuabili, a differenza dell'immagine *DIC* nel dataset A, la quale risulta molto simile allo sfondo. Durante lo studio dei dataset, si è notato che apportare piccole modifiche, come l'inversione dell'immagine o l'aumento del contrasto durante la fase di individuazione dei keypoint permette di ottenere allineamenti migliori nelle immagini di tipo *DIC.* In questo modo è possibile fronteggiare in maniera migliore il problema delle immagini simili all'immagine *DIC* del dataset A. Questo mette in evidenza come l'algoritmo SIFT sia un valido strumento per individuare punti chiave all'interno delle immagini. Come dimostrato da questo esperimento, anche in presenza di grandi spostamenti, l'algoritmo SIFT è in grado di individuare gli stessi punti chiave e consentire un allineamento corretto delle immagini. Ciò significa che, nonostante un'immagine sia soggetta a notevoli spostamenti, SIFT riesce comunque a individuare le caratteristiche chiave corrispondenti e allinearle in modo accurato.

# 6 Conclusioni e sviluppi futuri

Questa Tesi ha portato allo sviluppo della versione 1.0 del tool *DS4H Image Alignment*, disponibile gratuitamente come software open-source al link: *https://github.com/UniBoDS4H/IA*. In prima persona mi sono occupato dello studio del tool competitor, della gestione ottimizzata delle immagini e del progetto, ho realizzato le strutture per gli algoritmi automatici ed ho implementato questi all'interno del plugin, mi sono occupato del problema delle deformazioni elastiche tramite l'impiego del plugin *bUnwarpJ* ed infine ho realizzato uno script per creare release automatiche del codice per le piattaforme Windows, Mac e Linux.

*DS4H Image Alignment* è stato creato per soddisfare alcune richieste provenienti da Medici e Biologi che dovevano registrare immagini multimodali acquisite tramite microscopi a campo largo, al fine di condurre studi sulla colocalizzazione dei segnali intracellulari e migliorare l'analisi di provini istologici.

Il software sviluppato consente:

A) all'utente di eseguire vari tipi di allineamento, partendo da una forma manuale ad una completamente automatica:
   - L'allineamento manuale è realizzato mediante l'inserimento di un insieme di punti di riferimento (corner points) in ciascuna immagine. Questo consente l'utilizzo di specifiche trasformazioni geometriche per allineare le immagini caricate.
   - La funzionalità di allineamento automatico viene realizzata tramite la selezione di uno dei vari algoritmi implementati.

B) È inoltre possibile affrontare anche il problema della deformazione elastica utilizzando la libreria *bUnwarpJ;*

C) Il plugin sviluppato gestisce anche immagini multicanale;

D) Tutti gli allineamenti effettuati tramite questo software preservano le informazioni contenute in ogni immagine e possono essere utilizzati con tutti i tipi di immagini senza incorrere in errori.

In prima persona mi sono occupato dello studio del tool competitor, della gestione ottimizzata delle immagini e del progetto, ho realizzato le strutture per gli algoritmi automatici ed ho implementato questi all'interno del plugin, mi sono occupato del problema delle deformazioni elastiche tramite l'impiego del plugin *bUnwarpJ* ed infine ho realizzato uno script per creare release automatiche del codice per le piattaforme Windows, Mac e Linux.

In futuro, verrà sviluppata una libreria esterna per affrontare la deformazione elastica senza dipendere dal plugin *bUnwarpJ*, poiché questa libreria può causare la perdita di informazioni contenute in ogni immagine, tagliando anche sezioni importanti delle immagini.

In conclusione, sebbene *DS4H Image Alignment* sia ancora in fase di sviluppo, attualmente rappresenta la soluzione gratuita più completa per biologi e medici che necessitano di registrare immagini 2D ottenute tramite acquisizioni al microscopio. Questa versione del tool è stata presentata due contributi scientifici:

- *Poster Presentation, "DS4H Image Alignment (DS4H-IA), an open-source ImageJ/Fiji plugin for aligning multimodality 2D microscopy images."*, Filippo Piccinini, Fabio Vincenzi, Matteo Iorio, Marcella Tazzari, Maria Maddalena Tumedei, Jae-Chul Pyun, Enrico Giampieri, Giovanni Martinelli, Gastone Castellani, Antonella Carbonaro. DS4H Image Alignment (DS4H-IA), an open-source *ImageJ*/*Fiji* plugin for aligning multimodality 2D microscopy images. Straub Conference 2023, 25-26/05/2023, Biological Research Centre (BRC), Szeged, Hungary.

- *Abstract and Oral Presentation, "User-friendly open-source tools for aligning multimodality 2D microscopy images and performing single-cell co-localization analysis"*, Filippo Piccinini, Matteo Iorio, Fabio Vincenzi, Marco Edoardo Duma, Marcella Tazzari, Maria Maddalena Tumedei, Jae-Chul Pyun, Giovanni Martinelli, Gastone Castellani, Antonella Carbonaro, "User-friendly open-source tools for aligning multimodality 2D microscopy images and performing single-cell co-localization analysis". Cells & Extracellular Templates (CET) 2023, 7-9/06/2023, University Niccolò Cusano-Roma, Rome, Italy.

Inoltre, il tool verrà presto presentato alla comunità internazionale tramite una pubblicazione scientifica in rivista con Impact Factor, ma nel frattempo è gratuitamente disponibile online ed è quotidianamente utilizzato presso l'*Istituto Romagnolo per lo Studio dei Tumori "Dino Amadori"* IRCCS IRST di Meldola (FC, Italy).

# Bibliografia


[1] https://qima-lifesciences.com/en/sample-and-slide-preparation/,

[2] https://www.ncbi.nlm.nih.gov/books/NBK557663/,

[3] https://*ImageJ*.net/plugins/align-image-by-line-roi

[4] https://*ImageJ*.net/plugins/bigwarp

[5] https://www.ufz.de/index.php?en=47216

[6] https://icy.bioimageanalysis.org/plugin/ec-clem/

[7] https://imagej.net/plugins/elastix

[8] https://itk.org/

[9] https://*ImageJ*.net/plugins/linear-stack-alignment-with-sift

[10] https://*ImageJ*.net/plugins/register-virtual-stack-slices

[11] http://bigwww.epfl.ch/thevenaz/stackreg/

[12] https://*ImageJ*.net/plugins/trakem2/

[13] https://*ImageJ*.net/plugins/elastic-alignment-and-montage

[14] https://*ImageJ*.net/plugins/moving-least-squares

[15] https://*ImageJ*.net/plugins/image-stitching

[16] https://*ImageJ*.net/plugins/bigstitcher/

[17] https://pages.nist.gov/MIST/

[18] https://sourceforge.net/projects/micromos/

[19] Cos'è la Computer Vision? https://www.ibm.com/it-it/topics/computer-vision,

[20] Come si compila OpenCV per Linux?
https://docs.opencv.org/4.x/d7/d9f/tutorial_linux_install.html

[21] 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.

[22] 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.

[23] https://pyimagesearch.com/2015/03/16/image-pyramids-with-python-and-opencv/

[24] https://*ImageJ*.net/plugins/bunwarpj/

[25] https://en.wikipedia.org/wiki/PNG

[26] https://en.wikipedia.org/wiki/JPEG

[27] https://en.wikipedia.org/wiki/TIFF

[28] https://www.adobe.com/creativecloud/file-types/image/raster/tiff-file.html

[29] https://docs.opencv.org/3.4/javadoc/org/opencv/core/Mat.html

[30] https://*ImageJ*.nih.gov/ij/developer/api/ij/ij/process/ImageProcessor.html

[31] https://opencv.org/resources/media-kit/

[32] https://*ImageJ*.net/software/*ImageJ*/

[33] https://*ImageJ*.net/software/*Fiji*/

[34] https://en.wikipedia.org/wiki/Differential_interference_contrast_microscopy

[35] https://docs.opencv.org/4.x/da/df5/tutorial_py_sift_intro.html

# Ringraziamenti

Vorrei subito ringraziare la Prof.ssa Antonella Carbonaro dell'Università di Bologna per avermi seguito nel ruolo di relatrice durante la preparazione e lo svolgimento di questa Tesi, con prontezza, professionalità e disponibilità, dandomi l'opportunità di lavorare a questo progetto multidisciplinare.

Ringrazio il Prof. Giovanni Martinelli (permesso di utilizzo dati pazienti oncologici) per aver consentito l'utilizzo di dati acquisiti presso l'Ospedale IRST IRCCS Meldola (FC), ed il Prof. Gastone Castellani (accesso alle infrastrutture informatiche) per aver accesso alle infrastrutture con cui testare l'algoritmo sviluppato.

Desidero esprimere la mia gratitudine al Prof. Filippo Piccinini per avermi introdotto nel mondo della microscopia e dell'oncologia. Invece di essere semplicemente un correlatore, il Professor Piccinini si è distinto per la sua gentilezza e disponibilità, permettendomi di vivere un'esperienza eccezionale. È stato sempre pronto a discutere le mie soluzioni, offrendomi suggerimenti per migliorare il software. Ha sostenuto e incoraggiato le mie idee, facendomi sentire sempre competente in questo ruolo e credendo nelle mie potenzialità. Sono grato per tutto ciò che ha fatto per me.

Desidero esprimere un'infinita gratitudine alle persone più care che sono state sempre al mio fianco. In particolare, desidero ringraziare Zoffoli Sofia, Ferri Samuele, Strada Nicola, Romano Filippo e Vincenzi Fabio per il loro costante sostegno. Un ringraziamento speciale va anche ai miei amici di università Alessandro, Daniele, Ezmiron, Lorenzo, Luca, Stefano, Marco, Michele e Veri, per avermi permesso di passare le giornate di studio in maniera più spensierata. Vorrei esprimere la mia gratitudine nei confronti dei miei allenatori di pugilato, Gregori e Riccardo, per essermi stati sempre vicini e supportarmi costantemente. Inoltre, sono estremamente grato ad alcuni membri della palestra, tra cui Gabriele, Marco e Patrizio, i quali si sono dimostrati sempre disponibili ad offrirmi il loro aiuto e a farmi passare le giornate pre esami in maniera più spensierata. Il supporto di tutti e la vostra presenza costante hanno reso il mio percorso ancora più significativo. Vi ringrazio di cuore per tutto.

Vorrei esprimere la mia più sincera gratitudine alla mia adorata Ziuccia per il suo costante sostegno e i preziosi consigli che mi ha sempre donato lungo il mio percorso universitario. La sua presenza affettuosa e il suo incoraggiamento costante sono state una fonte di ispirazione e motivazione senza

pari. Non posso che ringraziarla di cuore per aver creduto in me e per avermi offerto un supporto incondizionato durante i momenti di dubbio e difficoltà. La sua generosità e la sua dedizione sono doni preziosi che porterò nel mio cuore per sempre.

Infine, desidero dedicare un profondo ringraziamento alle persone più care della mia vita, i miei genitori. Sono stati la mia luce guida, il sostegno costante che mi ha accompagnato in ogni passo del mio percorso nello studio. La loro presenza ha plasmato il mio essere, facendomi crescere e diventare un individuo migliore. Non esistono parole sufficienti per esprimere la gratitudine che provo nei loro confronti. Sono il mio punto di riferimento, e il loro incondizionato supporto ha reso possibile ogni mia conquista. Grazie a loro, ho imparato il valore del sacrificio e della determinazione. Sono profondamente grato per la loro presenza e per avermi aiutato a diventare la persona che sono oggi. Dedico questo momento ai miei genitori e soprattutto ai miei nonni e zii che nonostante abitino lontano da me sono sempre stati al mio fianco.