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: code restructuring seguendo modello MVC con ottimizzazione allineamento semiautomatico attraverso modulo gestione smart corners e rendering immagini per analisi di big-data

Tesi in: Programmazione

Relatore

Prof.ssa Antonella Carbonaro

Presentata da

Fabio Vincenzi

Correlatori

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

Anno Accademico 2023 / 2024

“– Senti, non importa quanto tempo ci vuole.

Non devi pensare troppo in là in questo lavoro, se no diventi matto.

– Allora a cosa devo pensare?

– A oggi. Guarda che bella giornata.”

Paolo Cognetti, Le otto montagne

Keywords

Istologia

Microscopia

Allineamento multimodale

Registrazione semiautomatica

ImageJ/*Fiji* plugin

Abstract (English version)

Often, in oncology studies, the analysis starts with histological samples, which are small tissue fragments taken from the patient. Specific portions of the sample are analyzed using selective dyes called markers. Sometimes, multiple markers are used to gather more information. However, it's not always possible to apply them simultaneously, and they are often used sequentially after washing the sample. The obtained images need to be aligned to proceed with single cell-based colocalization studies, simulating a simultaneous collection of the various intracellular signals. However, there is no standard procedure to align these images. Manual alignment is a time-consuming process prone to potential errors. The use of software could make the entire process faster and more reliable. Specifically, *DS4H Image Alignment* is an open-source plug-in developed for *ImageJ/Fiji* to align multimodal grayscale images. The first version of the tool offered an entry-level "Manual" mode that allows manual alignment of multiple images by specifying common reference points. Recently, we added an "Automatic" mode that enables alignment through an automated process of detecting common points. In this work, we (*a*) improved the full registration procedure, defining “smart corners” for converting the tool in a user-friendly software designed for life scientists, and (*b*) included several modules for obtaining a semi-automatic registration modality giving always an opportunity to quickly correct misalignments and always successful registering the input images.

Abstract (versione italiana)

Spesso, negli studi oncologici, si parte dall'analisi di campioni istologici, ossia di piccoli frammenti di tessuto prelevati dal paziente. Attraverso l'uso di coloranti selettivi chiamati marcatori, si analizzano specifiche porzioni del campione. Talvolta, per ottenere più informazioni, si utilizzano diversi marcatori. Tuttavia, non sempre è possibile applicarli contemporaneamente e spesso vengono utilizzati in sequenza dopo aver effettuato un lavaggio del campione. Le immagini ottenute devono quindi essere allineate in modo da poter procedere in studi di colocalizzazione a livello di singola cellula, simulando una raccolta simultanea dei vari segnali intracdllulari. Tuttavia, non esiste una procedura standard per allineare queste immagini. L'allineamento manuale è un processo lungo e soggetto a possibili errori. L'utilizzo di un software potrebbe rendere l'intero processo più rapido e affidabile. In particolare, *DS4H Image Alignment* è un plug-in open source sviluppato per *ImageJ/Fiji* che permette di allineare immagini multimodali a tonalità di grigio. La prima versione del software offriva una semplice modalità "Manuale", che consentiva di allineare manualmente più immagini specificando dei punti di riferimento comuni. Recentemente abbiamo aggiunto una modalità "Automatica" che permette l'allineamento delle immagini attraverso un processo automatizzato di individuazione dei punti comuni. In questo lavoro, abbiamo (*a*) migliorato l'intera procedura di registrazione, definendo "angoli intelligenti" per convertire lo strumento in un software di facile utilizzo progettato per gli scienziati della vita, e (*b*) incluso diversi moduli per ottenere una modalità di registrazione semi-automatica che dia sempre un'opportunità per correggere rapidamente disallineamenti disallineati e immagini che registrano sempre correttamente l'immagine in ingresso.

Indice

Elenco delle figure

1 Introduzione

Questo progetto di Tesi nasce all'interno del gruppo di ricerca "*Data Science for Health*" (*DS4H*), composto da ricercatori e professori dell'Università di Bologna (UniBo) e del "*IRCCS Istituto Romagnolo dei Tumori Dino Amadori*" (IRST) di Meldola (FC). Il gruppo *DS4H* nasce con lo scopo di coordinare professionisti e risorse sia dal settore informatico, dell'ingegneria dell'informazione e della fisica, sia dal settore biomedico, della biologia, chimica e medicina. In particolare, il progetto di tesi "*DS4H Image Alignment tool: code restructuring seguendo modello MVC con ottimizzazione allineamento semiautomatico attraverso modulo gestione smart corners e rendering immagini per analisi di big-data*" mira alla realizzazione di un plug-in di facile utilizzo per permettere l'allineamento di immagini in maniera rapida ed intuitiva. Il progetto ha l'obiettivo di consentire la registrazione di immagini multimodali acquisite con microscopi a campo largo per poter poi proseguire con studi di colocalizzazione di segnali intracellulari permettendo una miglior valutazione dei provini istologici in esame.

DS4H Image Alignment va a colmare alcune delle richieste formulate da Medici e Biologi di IRST. Per capire il contesto operativo, è necessario specificare come i professionisti di IRST procedono tipicamente nella valutazione dei tumori dei pazienti quando si ha a disposizione una biopsia del tessuto. Tipicamente si parte dalla realizzazione di provini istologici dai campioni di tessuto prelevati dal soggetto malato, a cui vengono applicati marcatori specifici per individuare cellule con particolari caratteristiche tumorali. Questi marcatori sono tipicamente delle sonde fluorescenti visibili tramite microscopia. Inoltre, per raccogliere più informazioni del campione si utilizzano spesso più marcatori. Tuttavia, questi non sempre possono essere applicati in parallelo e spesso vengono utilizzati in serie dopo un lavaggio del campione. A seguito della applicazione dei coloranti, vengono tipicamente ricavate delle immagini fluorescenti usando microscopi a campo largo. Le immagini così ottenute devono essere però allineate per poter procedere con studi di colocalizzazione, simulando una acquisizione in parallelo dei vari segnali (*i.e.* il nucleo, il citoplasma, la membrana). Purtroppo, non esiste una procedura standard per allineare le immagini così ottenute. L'allineamento manuale è tempo-dispendioso ed oggetto di possibili errori. Un software potrebbe rendere il tutto più rapido e affidabile, consentendo il riconoscimento delle aree comuni con conseguente allineamento dei segnali. Da questa necessità nasce quindi *DS4H Image Alignment*, con l'obiettivo di colmare questa mancanza tecnica e automatica nel mondo dell'istologia. Il risultato che si cerca di raggiungere è un'analisi rapida e precisa di provini istologici tramite uno strumento per l'allineamento delle immagini acquisite con diverse tecniche e colorazioni (multi-modalità).

Il riconoscimento del problema e le prime ipotesi di soluzione sono arrivati dalla Prof.ssa Antonella Carbonaro e dal Prof. Filippo Piccinini, concretizzatesi successivamente con la possibilità di utilizzare immagini ricavate da provini istologici di pazienti reali. Queste sono state rese disponibili dal Prof. Giovanni Martinelli, direttore scientifico dell'IRCCS IRST di Meldola, e dal Prof. Gastone Castellani, direttore della Scuola di Specializzazione in Fisica Medica dell'Università di Bologna.

In questo elaborato di Tesi verranno descritti i moduli implementati per realizzare un metodo di **registrazione semi-automatica** che unisce i punti di forza dei metodi manuali ed automatici implementati, dando all'utente una soluzione per la **registrazione veloce ed accurata** di qualsiasi sequenza di immagini di input, comprese immagini della categoria "big data" viste le importanti dimensioni raggiungibili ad esempio nel caso di mosaici di interi provini istologici. Attraverso la riorganizzazione completa del codice seguendo **modello MVC** ed attraverso la definizione di **"smart corners"** correlati da un insieme di features per l'editing e la gestione *DS4H Image Alignment* risulta ora essere un software estremamente user friendly e versatile per la soluzione del problema dell'allineamento di immagini multimodali di provini istologici,

Il progetto di tesi è stato svolto con la collaborazione di un altro tesista del Corso di Laurea Ingegneria e Scienze Informatiche, Matteo Iorio, con il quale ho quotidianamente lavorato per lo sviluppo del software e nello specifico dei seguenti task:

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.

Io sono responsabile e mio sono occupato in maniera più specifica dei punti A, C, D, I, J.

2 Provini istologici: importanza e preparazione

L'istologia, un settore della biologia, si dedica all'analisi della struttura microscopica e ultramicroscopica dei tessuti animali e vegetali. I campioni di questi tessuti vengono chiamati "provini istologici". In questo elaborato, il termine "provini istologici" si riferisce ai campioni prelevati da pazienti per monitorare e studiare patologie, identificare malattie e determinare il trattamento più adatto [1].

Nel corso degli anni, le tecniche di studio dei campioni istologici si sono perfezionate, con metodologie mirate a una migliore e più duratura conservazione dei campioni stessi, nonché a un'analisi più accurata. In particolare, la preparazione di un campione istologico prevede cinque fasi fondamentali:

- Fissazione
- Disidratazione
- Inclusione
- Sezionamento
- Colorazione

Durante la fissazione, vengono create condizioni simili a quelle fisiologiche per conservare il tessuto e rafforzare la struttura cellulare. Questo processo stabilisce una correlazione irreversibile tra le proteine. La fissazione può essere eseguita immergendo il campione in un materiale biologico come la formalina o in un fluido per il congelamento. Successivamente, il campione viene estratto per il sezionamento [2].

Segue la fase di disidratazione, che rimuove l'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 lavorato al microtomo. Vengono utilizzati alcoli di crescente intensità e volume per evitare danni alle cellule.

Successivamente, si procede all'inclusione, che consiste nella solidificazione del tessuto tramite l'uso di specifiche sostanze chimiche, di solito paraffina. Il campione viene esposto al calore e poi congelato per prevenirne il deterioramento. Poiché paraffina ed etanolo non sono miscibili tra loro, viene utilizzato un fluido miscibile con la paraffina per "pulire" il campione.

La fase successiva è il sezionamento, in cui il tessuto viene tagliato con un microtomo o un criostato (strumenti di precisione per il sezionamento). Vengono prodotte sezioni sufficientemente sottili, di solito di cinque micrometri, affinché la luce del microscopio possa attraversarle e consentirne l'analisi.

Infine, vengono applicate diverse colorazioni per evidenziare parti specifiche e proprietà del tessuto, aumentandone il contrasto e consentendo la distinzione da altri componenti del materiale biologico. Prima dell'applicazione delle colorazioni, il campione viene deparaffinato e reidratato, in modo che possa assorbire le tinture. Successivamente, il campione viene sciacquato e disidratato una volta completato il processo. Questa fase è nota come colorazione e può alterare la sezione sottoposta, richiedendo un lavaggio o una sostituzione prima di poter ripetere la procedura per studiare diverse parti utilizzando coloranti diversi[3][4].

Le figure seguenti mostrano i risultati ottenuti acquisendo immagini di un campione proveniente da una radice di bamboo utilizzando un microscopio a campo largo. Il campione è stato sottoposto a tutte le fasi descritte in precedenza. In particolare, sono state acquisite immagini di tipo Differential Interference Contrast (DIC) ed immagini fluorescenti utilizzando tre diverse sonde fluorescenti (Cy5, Cy3 e FITC).

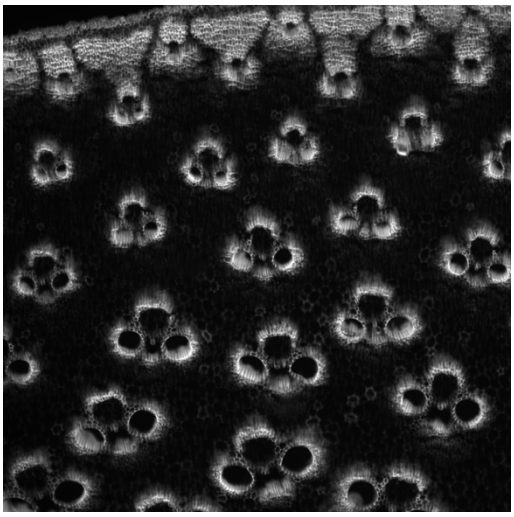

Figura 2.1: Radice di Bamboo con ingrandimento 10x (Cy3)

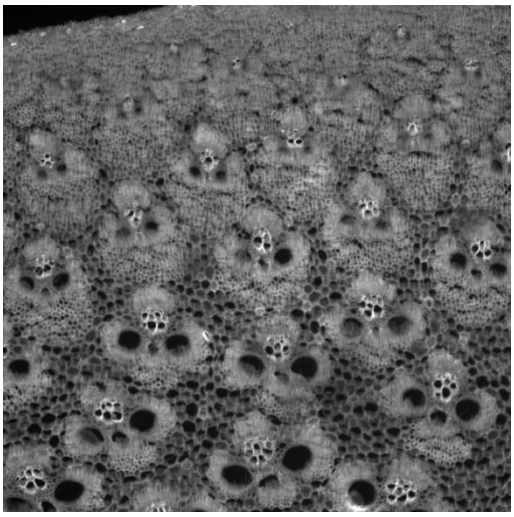

Figura 2.2: Radice di Bamboo con ingrandimento 10x (Cy5)

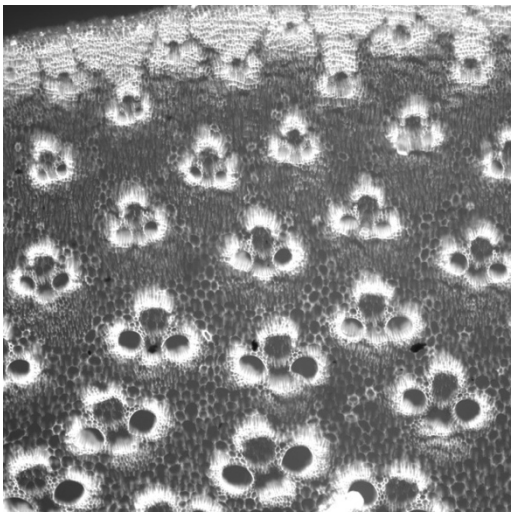

Figura 2.3: Radice di Bamboo con ingrandimento 10x (DIC)

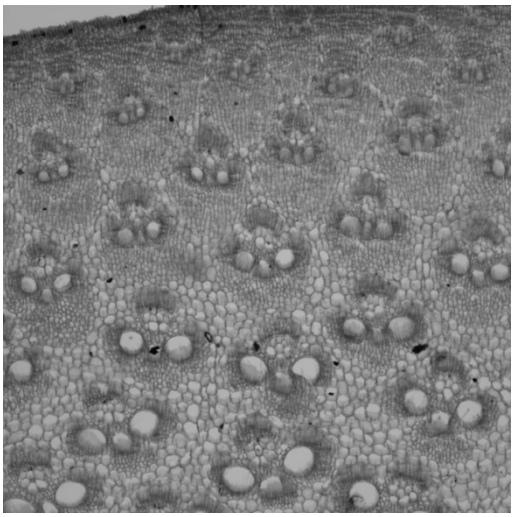

Figura 2.4: Radice di Bamboo con ingrandimento 10x (FITC)

3 ImageJ: Descrizione e Plugins

3.1 ImageJ

ImageJ è una suite di software per l'elaborazione delle immagini, originariamente sviluppata dall'NIH (National Institutes of Health) degli Stati Uniti nel 1997 come sostituto del precedente software *NIH Image*. Similmente ad altri programmi, *ImageJ* viene considerato [5] un *"Generalist image analysis tool"*, capace cioe` di ben adattarsi a un ampio set di problemi di natura diversa (come la diagnostica per immagini, la scienza dei materiali e la microscopia ottica); La sua flessibilità, riconosciuta come una delle chiavi del suo successo, è garantita da un framework che consente agli utenti di estendere facilmente le funzionalità del programma attraverso l'aggiunta di moduli sviluppati in linguaggi come *Java*, *Kotlin*, *Javascript*, *Scala*, *MATLAB*, e altri.

Essendo un software open-source supportato da una comunità eterogenea di ricercatori, biologi, sviluppatori e altri professionisti, esistono diverse distribuzioni parallele di *ImageJ*, comunemente chiamate "flavors", ognuna dedicata alla risoluzione di specifiche categorie di problemi. Accanto alla versione "classica" di *ImageJ* (*ImageJ* e *ImageJ2*), sono state sviluppate versioni specifiche per lo studio e l'analisi di immagini astronomiche chiamate *AstroImageJ*, o versioni che utilizzano l'engine *JavaFX* come interfaccia, cercando di offrire un'esperienza utente più intuitiva e aderente agli standard attuali dell'UX. Tuttavia, la build più utilizzata nell'ambito dell'analisi istologica è *Fiji*, che include una serie di plugin predefiniti e certificati in ogni versione. Data la sua ampia adozione e il suo continuo sviluppo, questo lavoro si concentrerà nello sviluppo di un applicativo compatibile con la versione *Fiji* di *ImageJ* [6]

3.2 ImageJ2

Il progetto "*ImageJ2*" è stato avviato nel 2009 ed è stato costruito su *SciJava Common*, che costituisce il nucleo centrale del progetto. È considerato il vero successore di *ImageJ* 1.x. La libreria è stata completamente riscritta, ripensata e ridisegnata per rendere più facile l'estensibilità a tutti i tipi di dati. Segue i principi SOLID separando le responsabilità e disaccoppiando l'interfaccia utente dal dominio applicativo. Questo approccio favorisce l'integrazione con diverse applicazioni e librerie, rendendo il progetto interoperabile.

Il progetto può essere considerato un framework condiviso per l'elaborazione delle immagini, che consente di scrivere il codice una sola volta e eseguirlo su diverse piattaforme come *KNIME*, *CellProfiler*, *OMERO* e *Icy*. Una delle principali caratteristiche è la compatibilità con la versione precedente, grazie alla libreria *ImageJ Legacy*, il che facilita la migrazione per gli sviluppatori. È inoltre dotato della libreria *ImageJ Updater*, che consente di aggiornare singolarmente i plugin della libreria *ImageJ Ops*. Quest'ultima include una vasta gamma di algoritmi di elaborazione delle immagini riutilizzabili, mentre *ImageJ Common* utilizza "*ImgLib2*" e il modello di dati delle immagini di base. Infine, per la maggior parte delle operazioni di input/output, il progetto fa uso della libreria SCIFIO ("SCientific Image Format Input and Output") [7] [8] [9].

3.3 Fiji

Fiji, acronimo di "Fiji Is Just ImageJ", è una distribuzione specifica di *ImageJ*, un software open-source per l'elaborazione delle immagini. *Fiji* è stata sviluppato per fornire agli utenti un'esperienza più completa e pronta all'uso, incorporando numerosi plugin, estensioni e strumenti aggiuntivi predefiniti. Una delle caratteristiche distintive di *Fiji* è la vasta collezione di plugin preinstallati, che coprono una vasta gamma di applicazioni e consentono di eseguire operazioni complesse di analisi e manipolazione delle immagini. Questi plugin sono sviluppati e mantenuti dalla comunità di utenti di *Fiji* e sono stati certificati per garantirne l'affidabilità e la qualità. *Fiji* è particolarmente popolare nell'ambito dell'analisi istologica, grazie alla disponibilità di plugin specifici per l'elaborazione delle immagini biomediche, la segmentazione cellulare, la morfologia e altre applicazioni comuni in questo campo. Inoltre, *Fiji* offre un'interfaccia utente intuitiva e personalizzabile, facilitando l'accesso alle funzionalità avanzate di *ImageJ*. Grazie alla sua natura open-source e alla vasta comunità di sviluppatori e utenti attivi, *Fiji* è in costante evoluzione e aggiornamento, con nuovi plugin che vengono costantemente sviluppati e con un supporto attivo da parte della comunità. Ciò contribuisce a rendere *Fiji* una delle distribuzioni di *ImageJ* più popolari e utilizzate nell'ambito dell'elaborazione delle immagini scientifiche [10].

3.4 Sviluppo di un plugin

Le funzionalità di *ImageJ* possono essere estese attraverso lo sviluppo di tre categorie di strumenti: macro, script e plugin [11].

- Le **macro** sono programmi che automatizzano una serie di operazioni di *ImageJ*. Sono relativamente semplici e limitati nel senso che possono eseguire solo operazioni già presenti

nel core di *ImageJ*. Le macro possono essere scritte utilizzando il linguaggio di scripting dedicato chiamato *IJM* (The ImageJ Macro Language) o registrate tramite strumenti appositi del programma.

- Gli **script** consentono l'esecuzione di porzioni di codice più complesse rispetto alle macro. *ImageJ* supporta l'esecuzione di script in vari linguaggi predefiniti come *Groovy*, *Python*, *JavaScript* e *Ruby*. Gli script possono essere eseguiti direttamente all'interno dell'istanza di *ImageJ* utilizzando l'Editor di Script. Tuttavia, a eccezione dei programmi scritti in *Clojure*, gli script sono interpretati a tempo di esecuzione, il che può sacrificare le prestazioni per compiti più impegnativi.
- I **plugin** offrono la massima flessibilità all'interno del framework di *ImageJ* e consentono di ottenere prestazioni di esecuzione paragonabili a quelle native. Possono includere interi moduli applicativi e sono essenziali per la creazione di nuove interfacce utente o per l'esecuzione di elaborazioni che richiedono calcoli su thread multipli.

DS4H Image Alignment è stato sviluppato come plugin, sfruttando appieno le potenzialità di estensione di *ImageJ*, consentendo di implementare tutte le funzionalità desiderate in modo efficiente e con prestazioni ottimali.

```
public class Image_Alignment implements PlugIn {
   public static void main(final String[] args) {
      new Image_Alignment().run(null);
   }

   @Override
   public void run(final String s) {
      try {
         new MainMenuGUI();
      } catch (Exception e) {
         throw new RuntimeException(e);
      }
   }
}
```

Figura 3.1: Semplice esempio di creazione plugin ImageJ

3.4.1 Strumenti necessari Per lo sviluppo di un plugin ImageJ

1. *Java SDK*: È indispensabile installare la versione più recente del *Source Development Kit* di *Java* poiché i plugin sono scritti in *Java*.
2. *Maven*: Poiché le operazioni di compilazione, installazione e testing devono essere eseguite tramite *Maven*, è necessario aggiungerlo al sistema operativo. In ambienti *Windows*, ciò può essere fatto aggiungendo l'eseguibile del programma alla variabile PATH.

Anche se concettualmente i plugin per *ImageJ* potrebbero essere sviluppati senza l'uso di IDE specifici, si consiglia vivamente di utilizzarli per la loro integrazione quasi universale con *Maven*. In questo modo, sarà semplificato l'utilizzo di comandi specifici per la compilazione, il testing e l'installazione. Il team ha scelto di utilizzare *IntelliJ Idea* di *JetBrains* [12] come strumento di sviluppo e compilazione, ritenendolo moderno e adatto alle esigenze richieste. Inizializzazione e compilazione Ecco una procedura riassuntiva per la creazione di un nuovo progetto compatibile con *SciJava* utilizzando *IntelliJ IDEA*:

1. Creare un nuovo progetto *Maven* tramite File → Project → *Maven*.
2. All'interno del file pom.xml, specificare le sezioni obbligatorie. Assicurarsi di inserire un *GroupId* che identifichi univocamente il progetto e un *ArtifactId* che specifichi il nome del file .jar generato ad ogni compilazione.
3. Creare una nuova classe di partenza del progetto implementando l'interfaccia specifica della tipologia di plugin desiderata.

Successivamente, sarà necessario configurare la modalità di debug dell'IDE. Per farlo, navigare nel menu Run → Edit Configurations e aggiungere una nuova voce Application con il valore di Main class corrispondente alla classe creata nel punto 3.

La procedura di compilazione per la distribuzione del plugin richiede l'utilizzo di *Maven*, in particolare il comando **mvn install**. Utilizzando *IntelliJ IDEA,* ecco una procedura riassuntiva per la distribuzione:

1. Aprire il progetto in *IntelliJ IDEA* e apportare le modifiche al codice necessarie.
2. Nel menu dedicato a *Maven*, eseguire l'operazione di installazione (install) e attendere il completamento dell'operazione. L'operazione avrà successo solo se non ci sono errori di compilazione. I file compilati saranno disponibili nella cartella **target** del progetto, oppure potranno essere trovati nella cartella di build predefinita di *Maven*: **UserHomeDirectory/.m2/repository/**.

3. Il plugin potrà essere installato in qualsiasi istanza di *ImageJ* trascinando il file .jar compilato all'interno della finestra dell'applicazione o utilizzando il menu Plugins → Install.

3.5 Maven

Maven è uno strumento di gestione dei progetti per Java che ha l'obiettivo di semplificare il processo di compilazione e gestione. *Maven* ha introdotto il concetto di *Project Object Model* (*POM*), un file *XML* che definisce la struttura, le dipendenze e altre informazioni essenziali del progetto.

Una delle caratteristiche principali di *Maven* è la sua capacità di gestire automaticamente le dipendenze del progetto. Utilizzando il *POM (Figura 3.2)*, Maven può scaricare e configurare le librerie esterne richieste dal progetto, semplificando notevolmente il processo di gestione delle dipendenze. Oltre alla gestione delle dipendenze e alla compilazione, *Maven* fornisce anche informazioni utili sul progetto. Attraverso il *POM* e le convenzioni di *Maven*, è possibile generare automaticamente informazioni sulle modifiche, report dei test unitari, report di copertura del codice e altro ancora. Queste informazioni aiutano gli sviluppatori a comprendere rapidamente lo stato del progetto e favoriscono l'adozione di buone pratiche di sviluppo [13].

Per creare un plugin per *ImageJ* o *Fiji*, è necessario utilizzare *Maven* come strumento di gestione delle dipendenze e compilazione del progetto. Quando si avvia un nuovo progetto *Maven*, una delle prime cose da fare è creare un file pom.xml appropriato dove definire le dipendenze del progetto, inclusi i moduli e le librerie di *ImageJ* o *Fiji* necessari per lo sviluppo del plugin. È importante includere correttamente queste dipendenze nel file pom.xml per garantire che il progetto abbia accesso alle risorse richieste.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns=http://maven.apache.org/POM/4.0.0 ... >
  <modelVersion>4.0.0</modelVersion>
  <parent>
    <groupId>org.scijava</groupId>
    <artifactId>pom-scijava</artifactId>
    <version>31.1.0</version>
    <relativePath />
  </parent>
  <name>DS4H Image Alignment</name>
  <description>
    An ImageJ plugin for Image Alignment...
  </description>
  <inceptionYear>2023</inceptionYear>
  <organization>
    <name>UNIBOData Science For Health</name>
    <url>https://github.com/UniBoDS4H/</url>
  </organization>
</project>
```

Figura 3.2: Versione parziale del pom.xml di DS4H-ImageAlignment

4 DS4H Image Alignment tool

DS4H Image Alignment è un plugin sviluppato per *ImageJ/Fiji*, che include una funzionalità di manuale attraverso la selezione di Corners comuni e un allineamento automatico ottimizzato. Il plugin supporta immagini RGB e offre la possibilità di riutilizzare direttamente lo stack allineato come nuovo input del plugin. Dettagli più approfonditi su queste caratteristiche saranno esaminati nei seguenti punti.

4.1 Introduzione generale:

Il plugin è dotato di un'interfaccia grafica che consiste in diverse finestre modali, che vengono visualizzate a seconda della funzionalità richiesta. Tra le principali notiamo:

- **Main Window:** Questa finestra rappresenta l'area dove viene visualizzato lo stack di immagini caricate, per ogni immagine è possibile attraverso un pulsante specificare se questa sarà il *target* dell'allineamento, è inoltre possibile rimuovere le immagini che non vogliamo più nello stack. È inoltre possibile accedere ad una serie di menu che permettono l'accesso a numerose 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 opzioni come "import", "export" o "clear".
- **Preview Window:** Questa finestra consente di visualizzare un'immagine singola e offre la possibilità di aggiungere o rimuovere i Corners per l'allineamento, fornendo la possibilità di accedere e modificare alcune impostazioni di visualizzazione di questi.
- **Output Window:** Questa finestra visualizza il risultato finale dell'allineamento e offre varie opzioni tra cui la possibilità di salvare le immagini allineate o una parte di esse, riutilizzarle per futuri allineamenti o applicare l'algoritmo di deformazione elastica *bUnwarpJ* a tutte le immagini.

Le funzionalità offerte da queste finestre consentono un'interazione intuitiva e efficace con il plugin *DS4H Image Alignment*, facilitando il processo di allineamento delle immagini e garantendo risultati ottimali.

4.2 Utilizzo di DS4H Image Alignment

Nei paragrafi successivi vengono mostrati i principali casi di utilizzo del plugin

4.2.1 Upload delle immagini

Una volta aperto il plugin da *ImageJ*/*Fiji* ci si ritroverà la Main Window *(Figura 4.1)* come prima finestra di interazione dalla quale attraverso il menu sarà possibile eseguire il caricamento di immagini singole o il caricamento di un progetto precedentemente salvato.

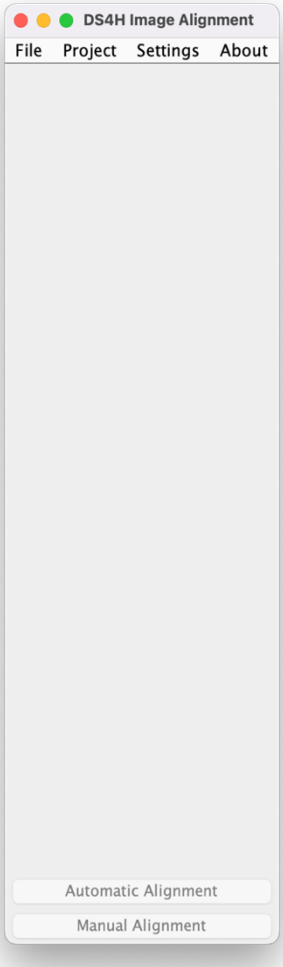

Figura 4.1: Main Window DS4H-ImageAlignment

4.2.1.1 Upload di immagini

Accedendo alla voce di menu *"File"* → *"Load images"* verrà aperto un *file chooser (Figura 4.2)* dal quale sarà possibile selezionare più immagini per effettuare il caricamento di esse all'interno del plugin.

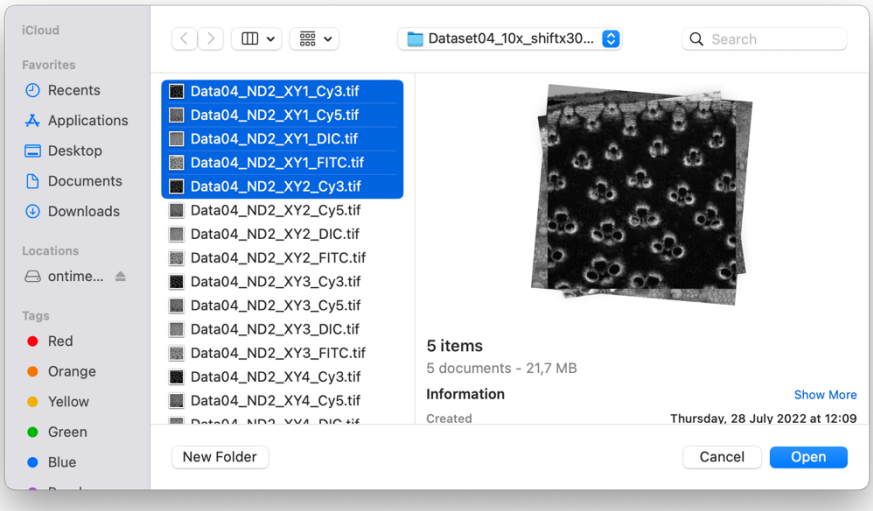

Figura 4.2: Selezione multipla di immagini tramite "File".

4.2.1.2 Upload di un progetto esistente

Accedendo alla voce di menu *"Project"* → *"Import"* si ha la possibilità di navigare tra le cartelle del file system per selezionarne una contenente un progetto precedentemente salvato, questo ci consentirà di recuperare tutte le immagini con i relativi Corner points selezionati.

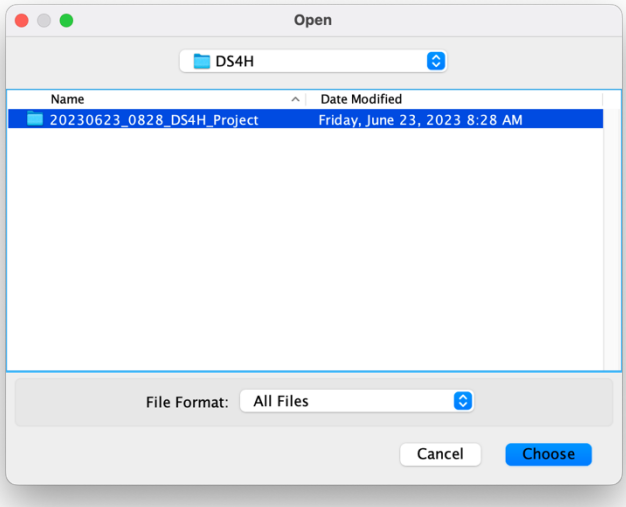

Figura 4.3: Caricamento di immagini tramite "Project > Import".

4.2.2 Settings

Accedendo alla voce di menu *"Settings"* si possono configurare diverse impostazioni del plugin. In particolare si possono modificare le impostazioni per l'allineamento *Automatico*, l'allineamento *Manuale* e la trasformazione elastica tramite *bUnwarpJ.*

4.2.2.1 Configurazione algoritmi di allineamento

Vengono fornite due finestre dalle quali è possibile modificare alcuni parametri di configurazione degli allineamenti, attraverso le configurazioni specificate viene permesso di capire cosa accade al momento dell'allineamento manuale o automatico.

- *Manual Settings:* Si potrà scegliere tra 3 differenti modelli di allineamento, si può successivamente decidere di abilitare la *"Traslazione"*, *"Scala"* e *"Rotazione"*. Infine viene offerta la possibilità di scegliere come gestire i punti in eccesso specificati sulle varie immagini. *(descrizione più approfondita capitolo 7.7)*
- *Automatic Settings:* Attraverso la voce "*Detector"* è possibile selezionare il tipo di algoritmo automatico,tra cui SURF[14] e SIFT[15], utilizzato per individuare i Corner, utilizzati per l'allineamento, in modo automatico, il numero di punti rilevati dal detector è influenzato dal *"Threshold factor*" che consente di includere anche punti considerati "non ottimi" nel processo di allineamento, più questo valore è alto più punti verranno considerati.
  Attraverso lo "*Scaling factor*" è possibile selezionare di quante volte ogni singola immagine deve essere scalata durante l'allineamento per garantire una più rapida gestione di immagini

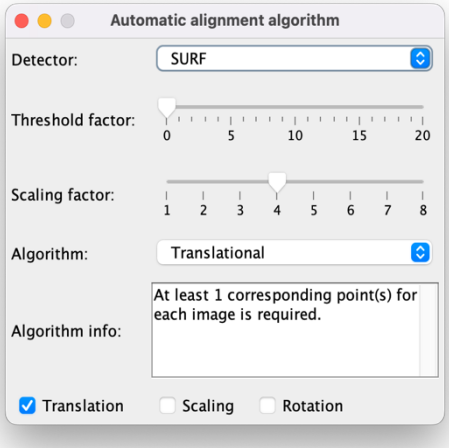

Figura 4.4: Finestra impostazioni allineamento automatico.

di grandi dimensioni. Vengono poi offerte le stesse configurazioni di selezione del modello di allineamento offerte dalle impostazioni *“manuali”*.

- *bUnwarpJ settings:* Attraverso questa interfaccia si possono configurare le impostazioni specifiche per la deformazione elastica delle immagini, Tra i numerosi parametri configurabili, uno dei più rilevanti è il parametro *"Mode"*, che consente di selezionare tra tre diverse opzioni:
    - *"Mono":* Questa modalità permette al programma di eseguire solo una registrazione unidirezionale, ovvero dallo sorgente alla destinazione. In questa modalità, l'allineamento avviene in una sola direzione, senza considerare eventuali discrepanze o differenze nella registrazione inversa.
    - "*Accurate":* Questa modalità prevede la registrazione bidirezionale, in cui l'allineamento viene eseguito sia dallo sorgente alla destinazione che dalla destinazione allo sorgente. L'utilizzo di questa modalità può richiedere più tempo di calcolo poiché il programma utilizza criteri di arresto più rigorosi per ottenere un allineamento più accurato.
    - *"Fast":* Questa modalità, simile alla modalità "Accurate", consente la registrazione bidirezionale, ma utilizza criteri di arresto meno rigidi. Ciò permette di ottenere risultati di allineamento più rapidamente, anche se potrebbe comportare una minore precisione rispetto alla modalità "Accurate".

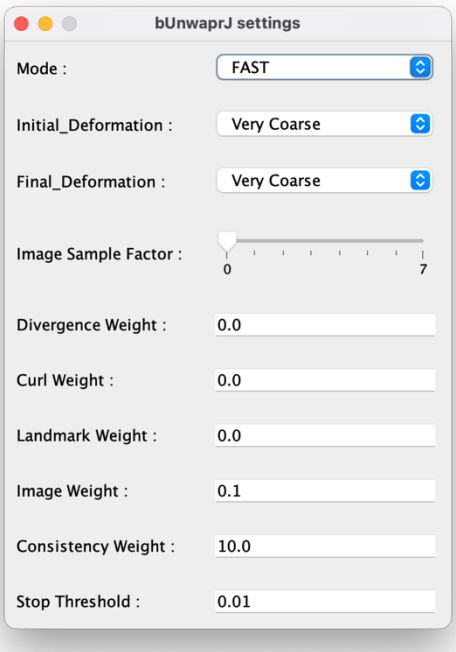

Figura 4.5: Menù di configurazione bUnwarpJ.

4.2.3 Target image

Tutti gli allineamenti proposti si basano sul modificare immagini sulla base di un immagine definita come *Target*, una volta caricate le immagini necessarie per l'allineamento è possibile attraverso il bottone *"TARGET"* andare a specificare quale tra le immagini dovrà essere utilizzata come riferimento per l'allineamento [16].

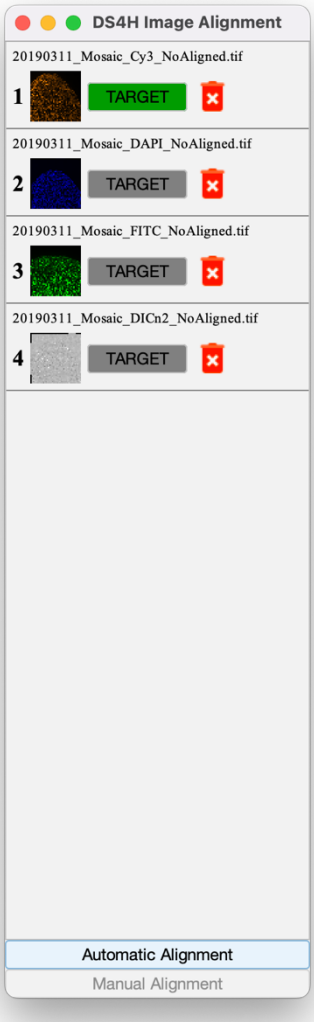

Figura 4.6: selezione Target image

4.2.4 Rimozione Immagini

È sempre possibile rimuovere un immagine dallo stack di allineamento tramite il bottone con l'icona del cestino rosso affianco al bottone *"TARGET"* una volta eliminata l'immagine questa non sarà più presa in considerazione durante gli allineamenti.

4.2.5 Gestione corner points

Si può effettuare l'inserimento di corner points per marcare punti salienti comuni alle immagini in questo modo l'algoritmo di allineamento manuale potrà utilizzare i suddetti punti per allineare le immagini facendo riferimento ai punti in comune con l'immagine target, la gestione dei Corner Points verrà discussa approfonditamente nel *capitolo 6*

4.2.6 Esportare un progetto

L'utente ha la possibilità di esportare l'intero progetto attraverso la voce di menu *"Project"* → *"Export"*. Questa funzione consente di salvare tutte le immagini insieme ai relativi Corners impostati su ciascuna di esse, nonché il target selezionato, in una cartella scelta dall'utente nel proprio sistema. Quando l'utente seleziona l'opzione "Export", viene visualizzata una finestra di dialogo che richiede all'utente di specificare il percorso di salvataggio del progetto. Automaticamente, verrà creata una cartella denominata "*YYYYMMDD_HHMM_DS4H_Project*" (con *YYYY* anno, *MM* mese, *DD* giorno, *HH* ora, *MM* minuto) che conterrà tutte le immagini e un file unico in formato JSON contenente tutte le informazioni relative al "target" e Corner Points specificati. In questo modo, l'utente può conservare e condividere facilmente il proprio progetto completo, inclusi i dati di allineamento, in un formato organizzato e portatile.

4.2.7 Allineamento Manuale

Attraverso la sezione bassa della finestra principale di *DS4H-ImageAlignemnt* è possibile effettuare l'allineamento manuale, il bottone verrà abilitato solamente se il numero di corner specificato su ogni immagine rispetta il numero minimo di corner richiesti dall'algoritmo di allineamento selezionato. Durante l'esecuzione dell'algoritmo viene mostrata una finestra di avanzamento grazie alla quale è possibile verificare lo stato di avanzamento dell'allineamento *(Figura 4.7)*, successivamente sarà possibile verificare l'output *(Figura 4.8)*.

4.2.8 Allineamento Automatico

È possibile effettuare un allineamento automatico il quale non prevede di specificare alcun Corner point nelle immagini, l'algoritmo sfrutterà il *Detector* specificato nelle impostazioni per rilevare le features più importanti tra le varie immagini per poi applicare la trasformazione desiderata ad esse. Come per l'allineamento manuale verrà mostrata durante l'esecuzione dell'algoritmo una finestra di avanzamento *(Figura 4.7) e successivamente l'output (Figura 4.8).*

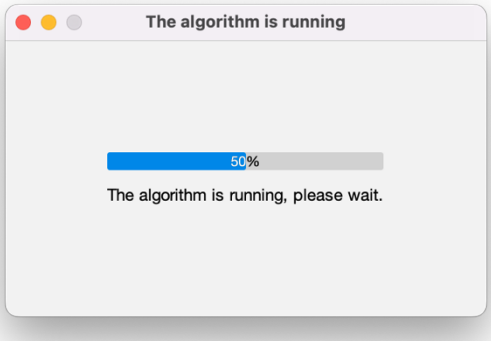

Figura 4.7: Finestra di loading.

Una volta terminato l'allineamento verrà visualizzata una finestra che mostra lo stack di immagini allineate *(Figura 4.8)*, lo stack fornito è uno stack completamente compatibile con gli stack di *ImageJ* quindi sarà possibile effettuare operazioni su di esso attraverso gli strumenti offerti da *ImageJ* stesso tra cui anche la creazione di uno Z-stack *(Figura 4.9).*

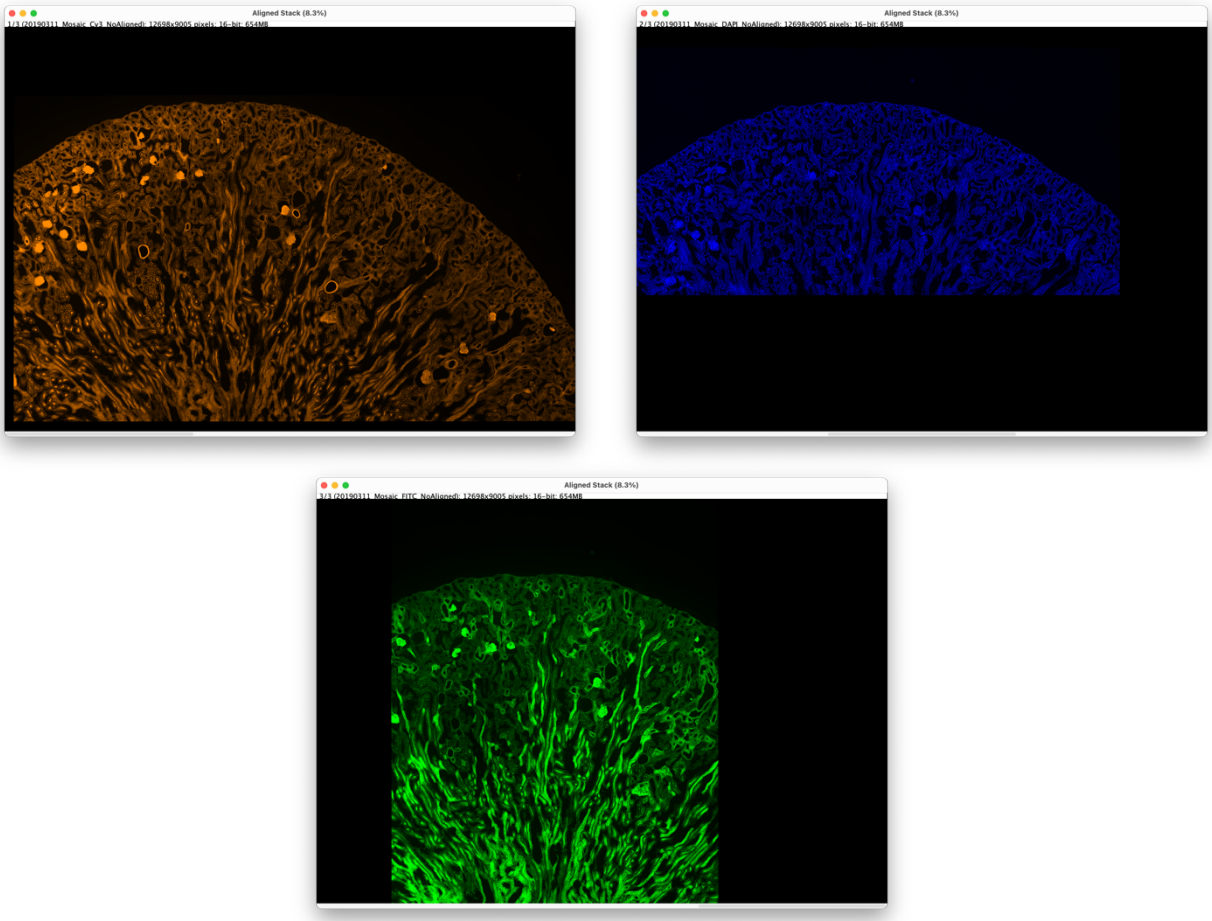

Figura 4.9: Output dell'allineamento

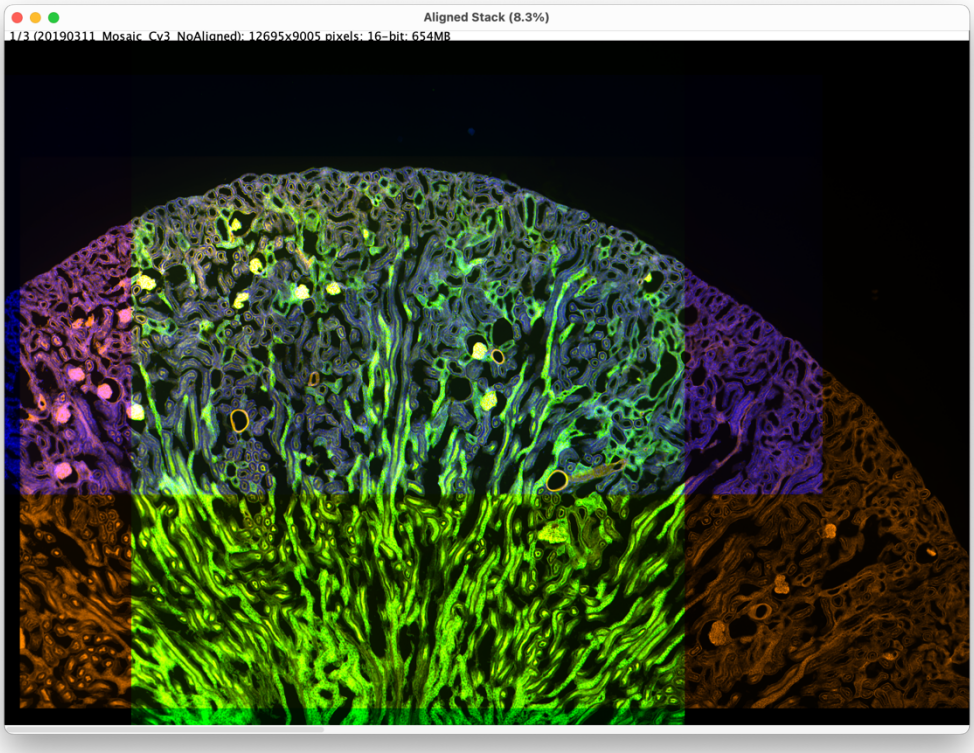

Figura 4.8: Creazione Z-Stack

4.2.9 Salvataggio delle immagini

Dalla schermata di output dell'allineamento è possibile salvare lo stack allineato, accedendo alla voce di menu *"Save"* → *"Save Project"* è possibile esportare tutte o parte delle immagini allineate. Verrà aperta una finestra *(Figura 4.10)* dalla quale sarà possibile selezionare le immagini da esportare, successivamente sarà necessario specificare il percorso di una cartella nella quale esportare le immagini, dove all'interno di essa verrà creata un ulteriore cartella denominata *"YYYYMMDD_HHMM_DS4H_AlignedImages"* contenente le immagini allineate.

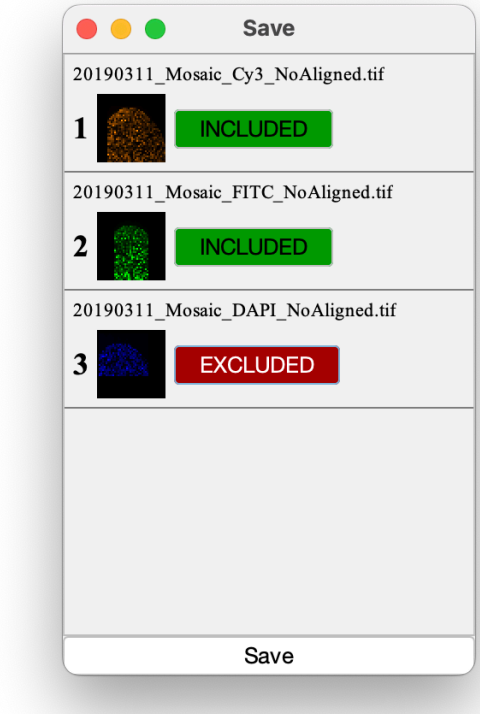

Figura 4.10: Finestra per il salvataggio delle immagini.

4.2.10 Reuse delle immagini

È possibile effettuare il reuse di tutte o solo alcune immagini dello stack allineato, ad esempio se non siamo soddisfatti dell'allineamento di una delle immagini, questo permette di effettuare il caricamento delle immagini allineate incluse, nella Main GUI dove successivamente possiamo ricaricare ulteriori immagini Applicazione algoritmo di deformazione elastica

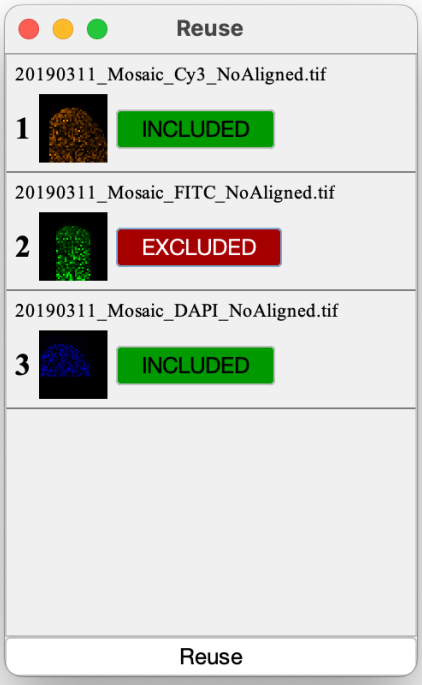

Figura 4.11: Finestra reuse immagini

4.2.11 Applicazione algoritmo di deformazione elastica

Una volta completato l'allineamento delle immagini, si può opzionalmente procedere con un ulteriore modifica delle immagini, ovvero l'applicazione dell'algoritmo di deformazione elastica fornito da *bUnwarpJ*. Attraverso il menu selezionare l'opzione *"Elastic"*. Attraverso i parametri di input inseriti nella finestra di configurazione di *bUnwarpJ* sarà possibile effettuare la trasformazione.

5 Code restructuring seguendo modello MVC

Per lo sviluppo di *DS4H-ImageAlignment* è stato seguito il pattern di progettazione Model-View-Controller (MVC). Il pattern MVC è un'architettura di progettazione software ampiamente utilizzata per separare la logica di business, la presentazione dei dati e l'interazione dell'utente in componenti distinti. Questo pattern offre numerosi vantaggi, tra cui una migliore modularità, manutenibilità, riusabilità del codice e testabilità.

5.1 Model

La sezione "model" del software si occupa dell'allineamento delle immagini ed è composta da diverse classi che collaborano tra loro per raggiungere questo obiettivo. Al centro di questo processo si trovano le classi **ImagePoints**, **Alignment** e **AlignedImage**.

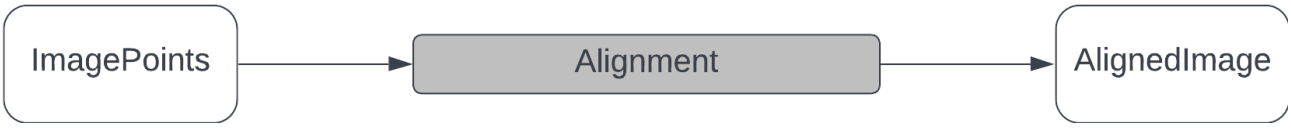

Figura 5.1: Processo di allineamento

La classe **ImagePoints** è una classe di fondamentale importanza in quanto rappresenta ogni immagine all'interno di *DS4H-ImageAlignment* permette inoltre di gestire un insieme di Corner points associati ad essa. Questi punti sono fondamentali per identificare le caratteristiche significative all'interno dell'immagine e vengono utilizzati come riferimento durante l'allineamento.

La classe fornisce metodi per gestire e manipolare i punti, come l'aggiunta, la rimozione e il conteggio.

La classe **Alignment** gestisce l'allineamento delle immagini nella maniera corretta a seconda dell'algoritmo e modalità selezionata. Questa classe coordina l'intero processo di allineamento, utilizzando diverse strategie e algoritmi specificati dall'utente. Essa gestisce la selezione dell'algoritmo corretto per l'allineamento, sia in modalità manuale che automatica. Inoltre, fornisce funzionalità per il monitoraggio dello stato di avanzamento dell'allineamento e per l'accesso alle immagini allineate.

La classe **AlignedImage** rappresenta un'immagine allineata. Contiene l'immagine stessa e il suo nome associato. Questa classe fornisce metodi per accedere all'immagine allineata e rilasciare le risorse associate ad essa quando non è più necessaria.

Oltre a queste classi centrali, ci sono altre componenti importanti.

La classe **PointDetector** è responsabile della rilevazione dei punti significativi all'interno delle immagini in maniera automatica attraverso l'utilizzo di specifici algoritmi, essa può essere configurata con diversi parametri per adattarsi alle specifiche esigenze dell'allineamento.

La classe **PointManager** gestisce l'insieme di immagini e punti utilizzati durante l'allineamento, tenendo traccia dell'immagine di riferimento e delle altre immagini da allineare. Infine, la classe **TargetImagePreprocessing** fornisce metodi per il preprocessing dell'immagine di riferimento prima dell'allineamento, per consentire all'immagine target di avere le dimensioni adeguate a sovrapporsi allo stack di immagini che verranno allineate su di essa.

Complessivamente, queste classi lavorano insieme per eseguire il processo di allineamento delle immagini. Utilizzando i punti significativi e gli algoritmi appropriati, l'allineamento viene eseguito in modo accurato e efficiente.

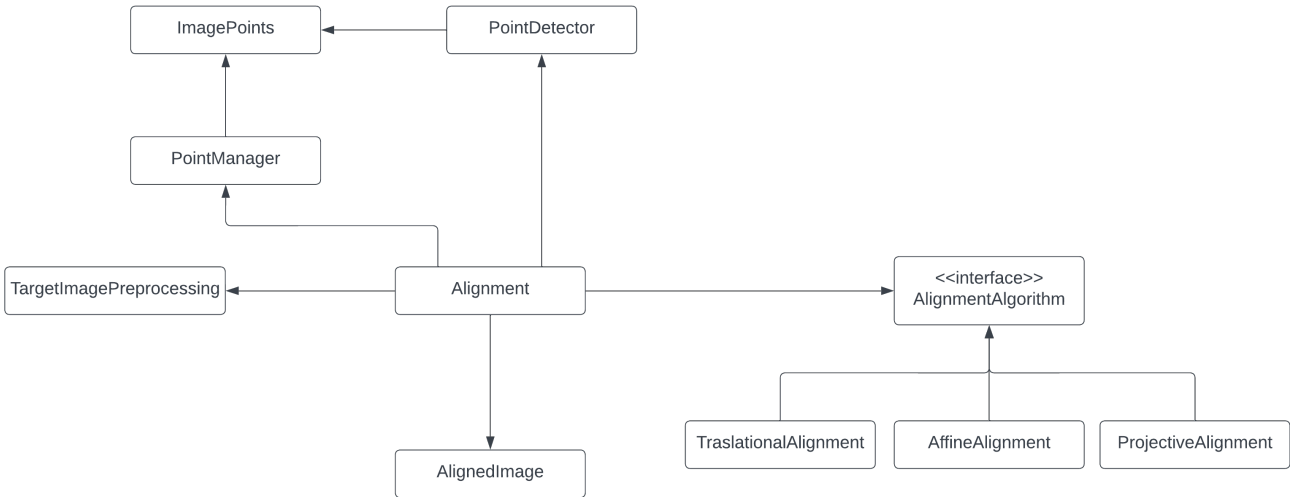

Figura 5.2: Diagramma UML dei model principali per allineamento

5.2 View

Il componente View si occupa della presentazione dei dati all'utente e dell'interfaccia grafica dell'applicazione. Di seguito riportiamo alcune classi significative.

MainGUI: rappresenta la schermata principale dell'applicativo dalla quale è possibile importare nuove immagini, impostare un'immagine come target, accedere alle varie pagine di configurazione degli algoritmi ed infine ci permette di avviare l'allineamento manuale o automatico delle immagini caricate.

PointSelectorGUI: rappresenta la finestra di selezione dei Corners estende la classe *ImageWindow* di *ImageJ* per semplificare la visualizzazione delle immagini e offrire un'integrazione completa con gli strumenti del software. Questo approccio consente di utilizzare le capacità di visualizzazione dell'immagine offerte da *ImageJ* all'interno della finestra di selezione dei Corners. Inoltre, l'integrazione completa con gli strumenti di *ImageJ* consente di utilizzare funzioni aggiuntive come l'applicazione di filtri, la misurazione delle dimensioni degli oggetti e l'annotazione delle immagini.

AlignmentOutputGUI: rappresenta un componente essenziale dell'applicazione poiché fornisce agli utenti una visualizzazione dell'output allineato. Essa è progettata per gestire stack di immagini allineate, consentendo agli utenti di esaminare e analizzare facilmente il risultato dell'allineamento.
Estendendo la classe *StackWindow* di *ImageJ*, la finestra AlignmentOutputGUI eredita tutte le funzionalità avanzate offerte da questa classe. La gestione semplificata dello stack consente agli utenti di scorrere tra le immagini allineate con facilità, regolare la luminosità e il contrasto e applicare operazioni su tutte le immagini simultaneamente.

StandardGUI: tutte le altre Finestre dell'applicativo implementano l'interfaccia *StandardGUI (Figura 5.3)*, la quale ci consente di implementare tre metodi per facilitare e unificare l'implementazione delle interfacce utente e la gestione dei suoi componenti.

```
public interface StandardGUI {
    public void showDialog();
    public void addListeners();
    public void addComponents();
}
```

Figura 5.3: Interfaccia StandardGUI per l'implementazione delle interfacce grafiche

5.3 Controller

Il componente Controller agisce come intermediario tra il Model e la View, gestendo il flusso di controllo e la gestione degli eventi. I Controller ricevono gli input dell'utente dalla View, eseguono le operazioni appropriate sul Model e aggiornano la View di conseguenza.

La classe **ImageController** svolge un ruolo centrale nella gestione dell'allineamento e della deformazione delle immagini nell'applicazione. Agisce come intermediario tra i modelli e le viste dell'applicazione. Le sue principali responsabilità sono:

1. Creazione di uno stack di immagini allineate: La classe **ImageController** crea uno stack di immagini allineate come risultato dell'allineamento e della deformazione. Questo stack può essere visualizzato o ulteriormente elaborato.
2. Recupero delle immagini allineate: La classe fornisce un metodo per ottenere l'elenco delle immagini allineate ottenute dall'allineamento o dalla deformazione delle immagini. Queste immagini verranno successivamente visualizzate come output dell'allineamento.
3. Controllo del flusso dell'applicazione: La classe **ImageController** gestisce lo stato corrente dell'applicazione, consentendo di controllare il flusso di esecuzione dell'allineamento e della deformazione delle immagini. Ad esempio, può impostare lo stato per avviare l'allineamento delle immagini o eseguire la deformazione elastica in base alle interazioni utente o alle condizioni specifiche.
4. Gestione delle risorse: La classe **ImageController** si occupa della gestione delle risorse legate alle immagini, come la liberazione delle immagini dall'heap per liberare memoria quando necessario.

6 Allineamento semiautomatico attraverso Smart Corners

Uno Smart Corner rappresenta un punto comune in più immagini. Grazie al posizionamento di essi su più immagini si ha la possibilità di effettuare un allineamento sfruttando i punti comuni specificati.

6.1 Posizionamento Smart Corners

Per inserire un Corner in un punto specifico di un'immagine sarà necessario effettuare doppio click nel punto desiderato, questo porterà alla creazione di un corner con relativo indice.

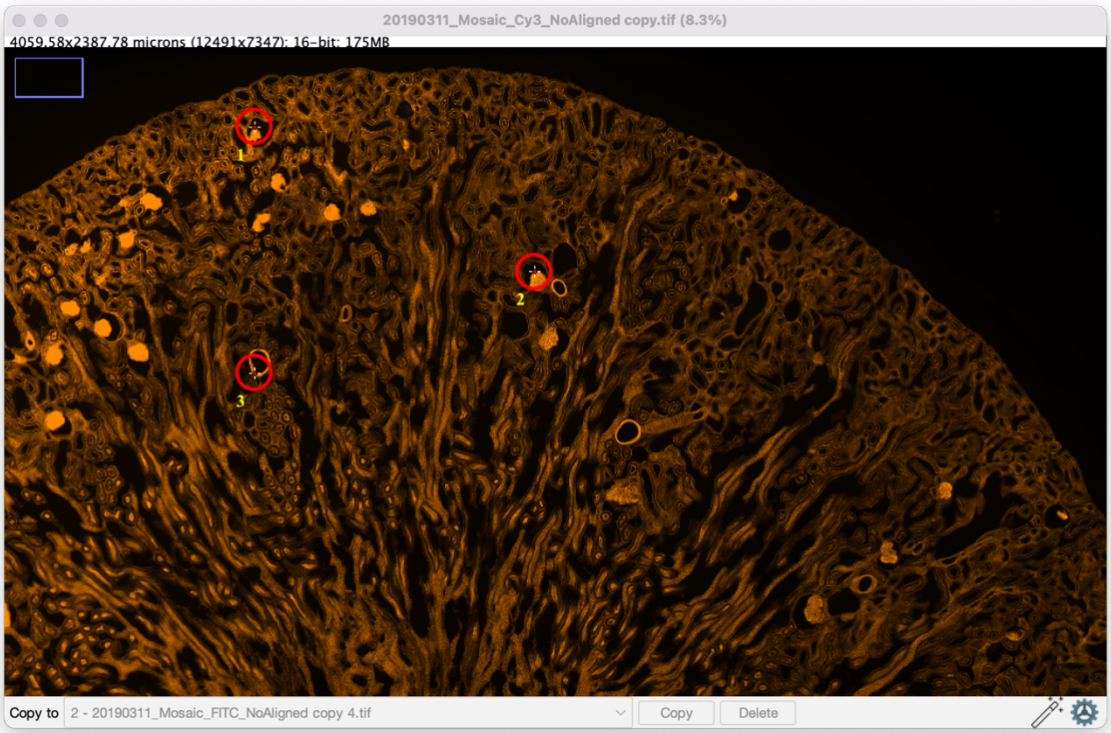

Figura 6.1: Finestra di inserimento Corners con 3 Smart Corners posizionati

6.2 Selezione Corners

Si può effettuare una selezione di più Corner premendo *ctrl* mentre si selezionano i punti scelti, i quali assumeranno un colore differente (predefinito e modificabile dalle impostazioni). Grazie alla selezione multipla possiamo effettuare operazioni simultaneamente su più corner in contemporanea.

6.3 Spostamento Corners

Una volta posizionato un Corner si può facilmente modificare la sua posizione effettuando un Drag and Drop di esso nel punto desiderato è anche possibile il movimento di un determinato gruppo di Corner effettuando una selezione di essi, questo permette di mantenere le distanze tra i punti.

6.4 Rimozione Corners

È poi possibile rimuovere un Corner o una selezione di più Corner, basterà cliccare *Delete* dalla barra degli strumenti in basso dopo aver effettuato una selezione dei Corner che si vogliono cancellare.

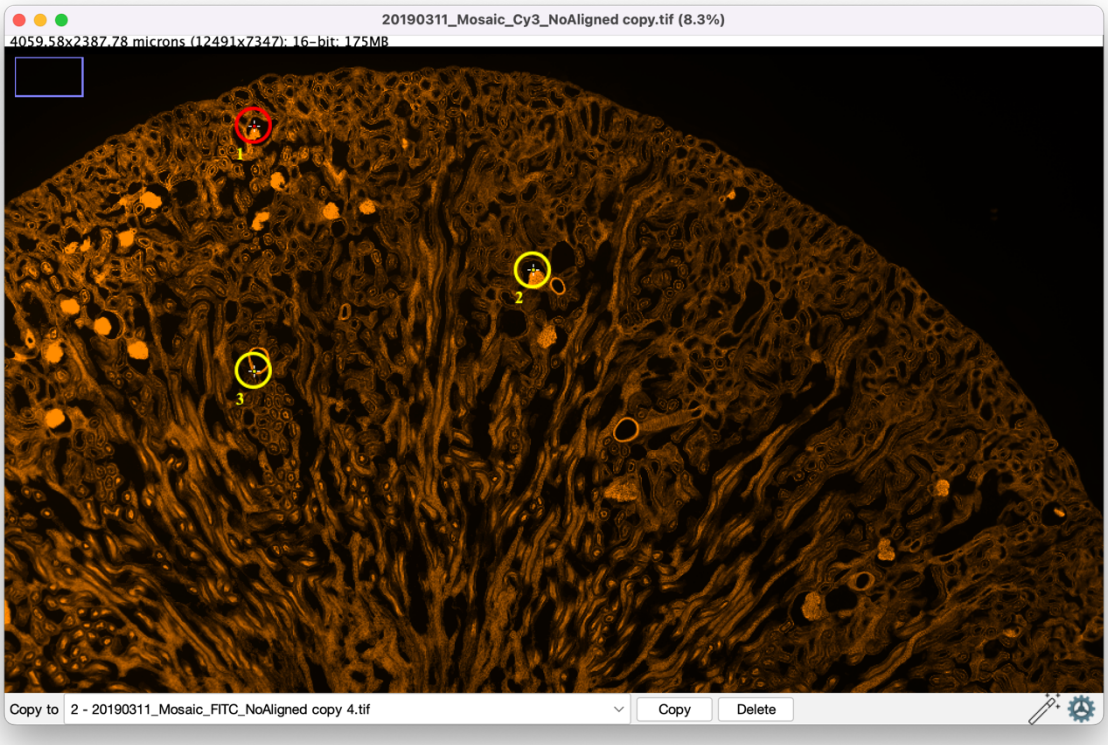

Figura 6.2: Finestra di inserimento Corners con 2 Smart Corners selezionati

6.5 Copia Corners

Si può copiare uno o più Corner da un'immagine ad un'altra selezionando il gruppo di Corner del quale vogliamo fare una copia e scegliendo l'immagine nella quale andremo ad aggiungere i Corner attraverso la combo box, successivamente premendo *Copy* i Corner selezionati vengono copiati

nell'immagine specificata. Se uno o più corner risulterebbero fuori dall'immagine selezionata questi non vengono copiati

Figura 6.3: Messaggio che ci indica che alcuni dei Corner non sono stati copiati

6.6 Settings Corners

Attraverso il bottone "Settings" in basso a destra si potrà accedere alla finestra di impostazioni dei Corner, dalla quale sarà possibile modificare il colore di essi, il colore della selezione, il colore dell'indice, è inoltre possibile modificare la dimensione dei Corner ed infine si può effettuare un cambio di indice per un Corner specificato.

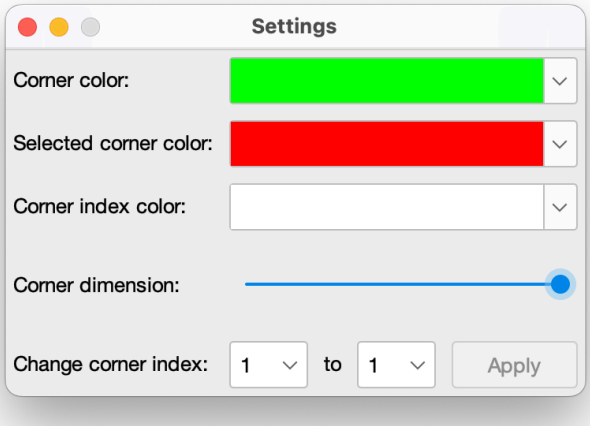

Figura 6.4: Finestra impostazioni Corners.

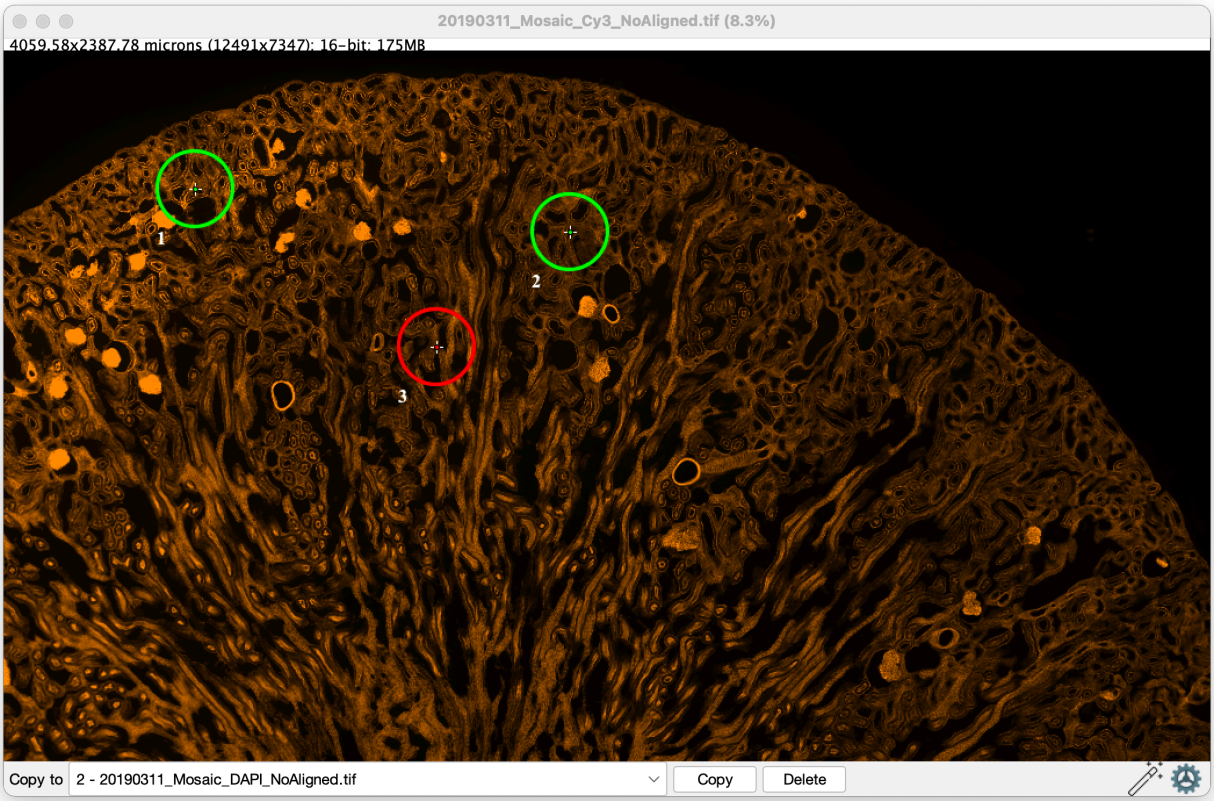

Figura 6.5: Corner con impostazioni di visualizzazione modificate

6.7 Allineamento

Dopo aver inserito in ogni immagine un numero di Smart Corner sufficienti per eseguire l'allineamento selezionato, si può eseguire l'allineamento manuale premendo il bottone *"Manual Alignment"*, che andrà a generare uno stack di immagini corrispondenti alle originali allineate in base ai punti specificati in precedenza.

6.7.1 Allineamenti manuali

L'utilizzo di un algoritmo di co-registrazione richiede l'applicazione di un modello di trasformazione specifico tra la Sorgente e il Template. Questo modello rappresenta una funzione che sposta i punti di un'immagine rispetto a un sistema di riferimento. Esistono due tipi principali di trasformazioni: "rigide" ed "elastiche". Le trasformazioni rigide rientrano nella categoria delle isometrie, ovvero delle

trasformazioni geometriche che mantengono invariate le lunghezze, gli angoli e le aree dell'immagine di riferimento. Queste operazioni sono descritte da una singola equazione applicata all'intera immagine e sono considerate più concise e immediate.

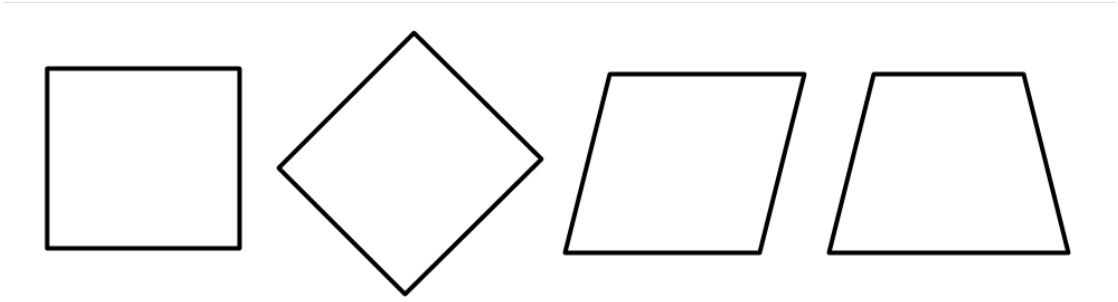

Figura 6.6: Modelli globali. Da sinistra a destra: Immagine originale, Rigida, Affine e Proiettiva.

D'altra parte, le trasformazioni elastiche sono rappresentate da un insieme di equazioni, ognuna dedicata a una specifica regione dell'immagine di riferimento. Queste trasformazioni possono essere immaginate come la deformazione di un corpo elastico soggetto a diverse forze esterne, ciascuna associata a un'equazione diversa. A causa del loro modo particolare di essere applicate, queste trasformazioni sono anche definite "globali" o "locali". Tuttavia, dato che *DS4H Image Alignment* si occupa principalmente di oggetti rigidi, questo progetto di tesi si concentrerà solo sulle trasformazioni globali, considerate più appropriate ed efficienti per lo scopo. A tale scopo, saranno esaminati i principali modelli presentati nella letteratura [17]:

6.7.2 Modello rigido

Le isometrie (o movimenti rigidi) sono trasformazioni del piano che mantengono invariate le distanze tra i punti dell'immagine traslata. In altre parole, prendendo $\overline{AB}$ come la distanza tra i punti A e B e $\overline{A'B'}$ come la distanza tra i punti trasformati, si avrà allora che: $\overline{AB} = \overline{A'B'}$

Ne consegue che tutte le operazioni isometriche trasformano un'immagine in una figura congruente, cambiandone solo la posizione sul piano. Concettualmente, la trasformazione rigida può essere descritta come l'applicazione di un'operazione di traslazione + rotazione; considerando $p = [x, y]^T$ un qualsiasi punto dell'immagine s e $p' = [x' , y']^T$ il medesimo punto posto nell'immagine t, allora le coordinate cartesiane delle due osservazioni saranno legate tramite l'equazione:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta & a \\ \sin\theta & \cos\theta & b \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

Nell'equazione, θ rappresenta l'angolo di rotazione dell'immagine da applicare, mentre la coppia di valori a e b indicano la traslazione lungo l'asse delle ascisse e delle ordinate rispettivamente. Poiché questo modello è costituito da un numero limitato di parametri, è adatto solo in casi limitati in cui due sequenze di immagini differiscono solo per rotazione e traslazione. Tuttavia, la registrazione di immagini microscopiche digitali tramite questo modello può risultare appropriata, poiché il dispositivo di acquisizione rimane sempre perpendicolare e costantemente distante dai campioni. Pertanto, in questo contesto specifico, l'utilizzo di tale modello può fornire risultati adeguati.

6.7.3 Modello affine

Il modello affine può essere descritto come una combinazione di trasformazioni di similarità e l'applicazione di shearing sull'immagine. In particolare, esso include l'utilizzo di funzioni come Identità, Traslazione, Scala, Omotetia, Similarità, Riflessione, Composizione e Shearing, in qualsiasi ordine e sequenza.

Dal punto di vista visivo, questo tipo di trasformazione può essere immaginato come l'allungamento di un oggetto parallelamente al suo piano dopo l'applicazione di una forza sui suoi bordi. Pertanto, una caratteristica peculiare di questo modello è che mantiene tutte le linee che erano parallele prima della sua applicazione. Tuttavia, a differenza del modello rigido, le affinità non garantiscono la conservazione degli angoli e delle distanze tra i punti.

Basandosi sulla matrice introdotta con il modello rigido, il modello affine è legato dall'equazione:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{A}\cos\theta & B\ \sin\theta & a \\ C\ \sin\theta & \mathrm{D}\cos\theta & b \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

dove *A*, *B*, *C*, *D* sono numeri reali. Si può dimostrare che ogni operazione di affinità gode delle seguenti proprietà:

- Collinearità tra i punti: se tre punti P, Q, R sono allineati, i loro corrispondenti in un'affinità P', Q', R' sono anch'essi allineati.
- Parallelismo: a rette parallele corrispondono rette parallele.

- Rapporto dei segmenti paralleli: il punto medio di un segmento corrisponde sempre al punto medio di un segmento omologo.
- se la figura S' è l'immagine corrispondente di una figura S, allora $\frac{Area(S')}{Area(S)} = |det\ A|$, dove $det\ A = ad - bc$

6.7.4 Modello Proiettivo

La trasformazione proiettiva, conosciuta anche come omografica o prospettica, è una delle tecniche più utilizzate nel campo della registrazione delle immagini ed è considerata una delle tipologie più comuni di trasformazione lineare. In poche parole, questa operazione può essere immaginata come il posizionamento di un occhio nello spazio. Di conseguenza, le dimensioni degli oggetti non saranno misurabili (poiché non ci saranno informazioni sulla loro profondità) e l'orizzonte verrà considerato parte integrante del piano.

Dal punto di vista pratico, una delle principali differenze rispetto alle trasformazioni affini è che la trasformazione proiettiva non garantisce il mantenimento del parallelismo delle linee o del rapporto tra gli elementi dell'immagine.

Una tipica trasformazione omografica assume una forma matriciale di questo tipo:

$$\begin{bmatrix} x_t \\ y_t \\ 1 \end{bmatrix} = \begin{bmatrix} \mathrm{A}\cos\theta & B\ \sin\theta & a \\ C\ \sin\theta & \mathrm{D}\cos\theta & b \\ G & H & 1 \end{bmatrix} = \begin{bmatrix} x_s \\ y_s \\ 1 \end{bmatrix}$$

Rimangono validi i formalismi già precedentemente introdotti per il modello affine.

6.8 Implementazione degli algoritmi

- **Traslativo:** Richiede almeno un Corner corrispondente tra le immagini per determinare la traslazione necessaria per allinearle. Questo algoritmo viene utilizzato quando le immagini hanno solo uno spostamento di posizione senza alcuna deformazione.
- **Affine:** L'algoritmo di allineamento affine richiede almeno tre Corner corrispondenti tra le immagini. Utilizzando questi punti, l'algoritmo calcola i parametri di una trasformazione

affine, che comprende traslazione, rotazione, ridimensionamento e distorsione affine (shering).

- **Proiettivo:** L'algoritmo di allineamento proiettivo richiede almeno quattro Corner corrispondenti tra le immagini. Utilizzando questi punti, l'algoritmo calcola i parametri di una trasformazione proiettiva, nota anche come omografia. Questo tipo di allineamento è in grado di gestire deformazioni prospettiche complesse, come ad esempio l'effetto di una telecamera inclinata o un punto di vista obliquo.

Per l'algoritmo Traslativo sarà possibile abilitare o disabilitare Traslazione, Scala e Rotazione.

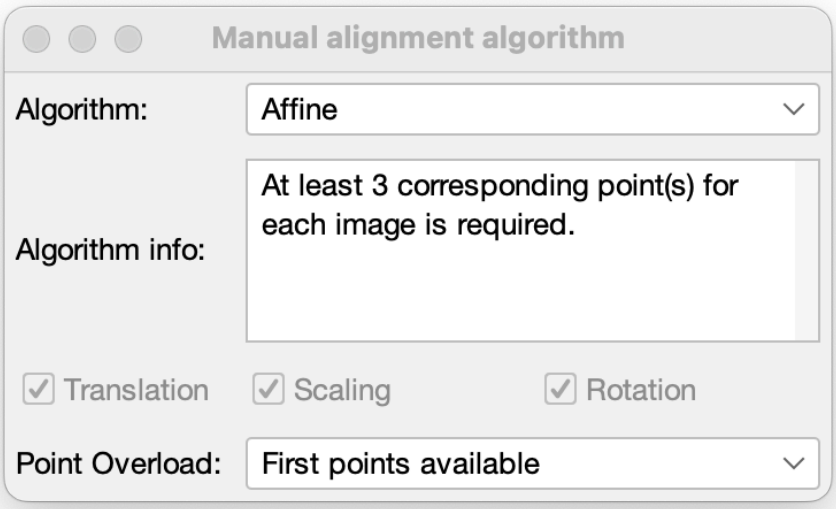

Figura 6.7: Finestra di impostazioni allineamento manuale.

```
public interface AlignmentAlgorithm {

   AlignedImage align(final ImagePoints targetImage, final ImagePoints imageToShift, final ImageProcessor ip)
throws IllegalArgumentException;

   Mat getTransformationMatrix(final MatOfPoint2f srcPoints, final MatOfPoint2f dstPoints);

   void transform(final Mat source, final Mat destination, Mat H);

   int getLowerBound();

   void setPointOverload(PointOverloadEnum overload);

   PointOverloadEnum getPointOverload();
}
```

Figura 6.8: Interfaccia che implementano gli algoritmi di allineamento.

Ogni algoritmo di allineamento implementa questa interfaccia, che definisce i seguenti metodi:

- **align**: esegue l'allineamento effettivo dell'immagine *"imageToShift"* su *"targetImage"*, utilizzando i punti di corrispondenza delle due immagini. L'implementazione di questo metodo deve gestire eventuali eccezioni di tipo **IllegalArgumentException**.
- **getTransformationMatrix**: calcola e restituisce la matrice di trasformazione da utilizzare durante l'allineamento in base alle corrispondenze dei punti.
- **transform**: applica la matrice di trasformazione *"H"* per modificare l'immagine di origine *"source"* e salva il risultato nell'immagine di destinazione *"destination"*.
- **getLowerBound**: restituisce il numero minimo di punti necessari per eseguire l'allineamento con l'algoritmo selezionato.
- **setPointOverload** e **getPointOverload**: impostano e restituiscono l'overload specificato per l'algoritmo, rappresentato dall'enumerazione **PointOverloadEnum**.

È importante notare che il metodo **getTransformationMatrix** deve tenere conto dell'overload dei punti e considerare correttamente i punti in base al tipo di overload scelto.

6.9 Gestione numero eccessivo di Corner

Se vengono selezionati un numero di Corner superiore a quelli necessari per l'allineamento selezionato si può scegliere di gestirli in 3 modi differenti:

- **First available *(default)*:** vengono utilizzati i primi Corner utili per l'allineamento, i successivi non vengono considerati
- **RANSAC:** permette di identificare i punti corrispondenti tra le immagini, anche in presenza di punti errati o non corrispondenti. L'algoritmo seleziona casualmente un sottoinsieme di punti e stimando i parametri del modello. Successivamente, viene verificato quali altri punti si adattano al modello stimato entro un certo limite di errore. Questo processo viene ripetuto per un numero fisso di iterazioni. Infine, viene restituito il modello con il miglior set di punti. Un esempio semplice è il "fitting" di una retta su un insieme di dati *(Figura 6.9)* [18] , che consiste nella costruzione di una funzione matematica che rappresenti in modo appropriato tali dati. Questo processo può essere affrontato utilizzando il metodo dei minimi quadrati, che cerca di trovare una retta ottimale che si adatti a tutti i punti dell'insieme, inclusi gli outlier (ovvero i punti che "non sono vicini" alla retta). Tuttavia, questo approccio può portare a una retta che non rappresenta adeguatamente gli inlier (i punti che possono essere rappresentati dalla retta). Per affrontare questo problema, si può utilizzare l'algoritmo RANSAC. RANSAC è in grado di produrre un modello calcolato solo sui punti inlier, senza considerare gli outlier. Questo metodo funziona correttamente a patto che la probabilità di selezionare solo punti inlier dai dati sia sufficientemente alta [19].

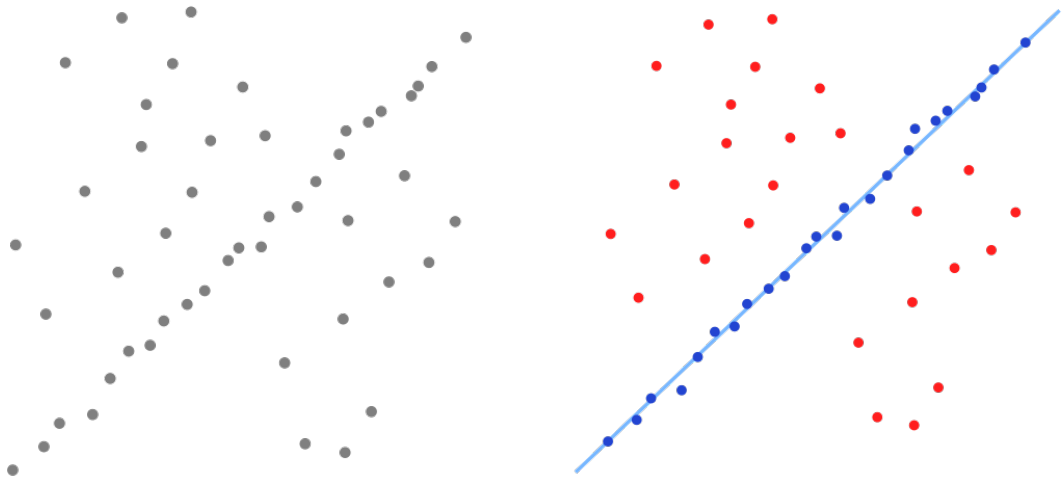

Figura 6.9: Stima di una retta con RANSAC dove gli outlier non influenzano il risultato

- **Minimum Least Squares:** calcola la trasformazione geometrica migliore tra i punti corrispondenti. Questo metodo cerca di minimizzare la somma dei quadrati delle differenze tra i punti corrispondenti trasformati e i punti corrispondenti originali [20].

```
private Point minimumLeastSquare(final Point[] srcArray, final Point[]
dstArray){
    final double[] deltaX = new double[srcArray.length];
    final double[] deltaY = new double[srcArray.length];

    IntStream.range(0, srcArray.length).parallel().forEach(i -> {
        deltaX[i] = dstArray[i].x - srcArray[i].x;
        deltaY[i] = dstArray[i].y - srcArray[i].y;
    });

    final double meanDeltaX = Core.mean(new MatOfDouble(deltaX)).val[0];
    final double meanDeltaY = Core.mean(new MatOfDouble(deltaY)).val[0];
    return new Point(meanDeltaX, meanDeltaY);
}
```

Figura 6.11: Implementazione Minimum Last Squares

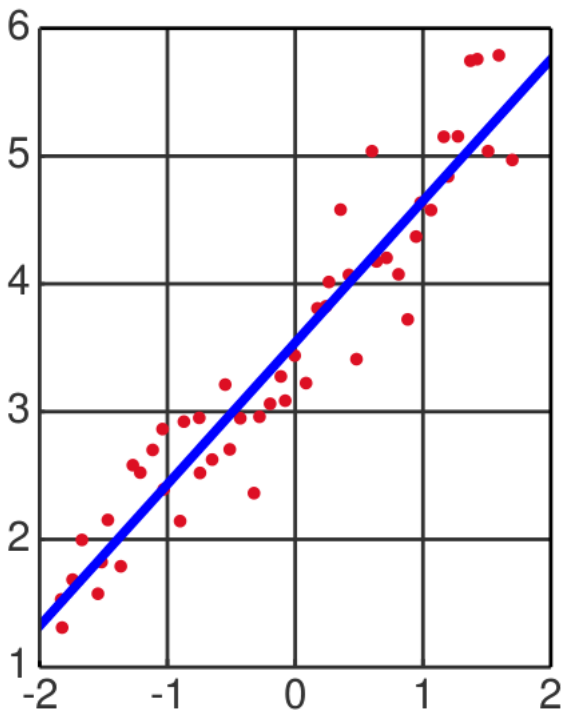

Figura 6.10: Individuazione di una retta tramite Minimum Least Squares

6.10 Preprocess dell'immagine target

Durante lo sviluppo del plugin è nata la necessità di capire con quali dimensioni ogni immagini sarebbe dovuta apparire in output e ci siamo accorti che non sarebbe bastato un semplice allineamento tra l'immagine target e i vari riferimenti, in quanto questo avrebbe portato alla perdita di alcuni pixel, il risultato che volevamo ottenere in output erano delle immagini tutte con la medesima dimensione ma senza aver perso dei dati *(Figura 6.9)*.

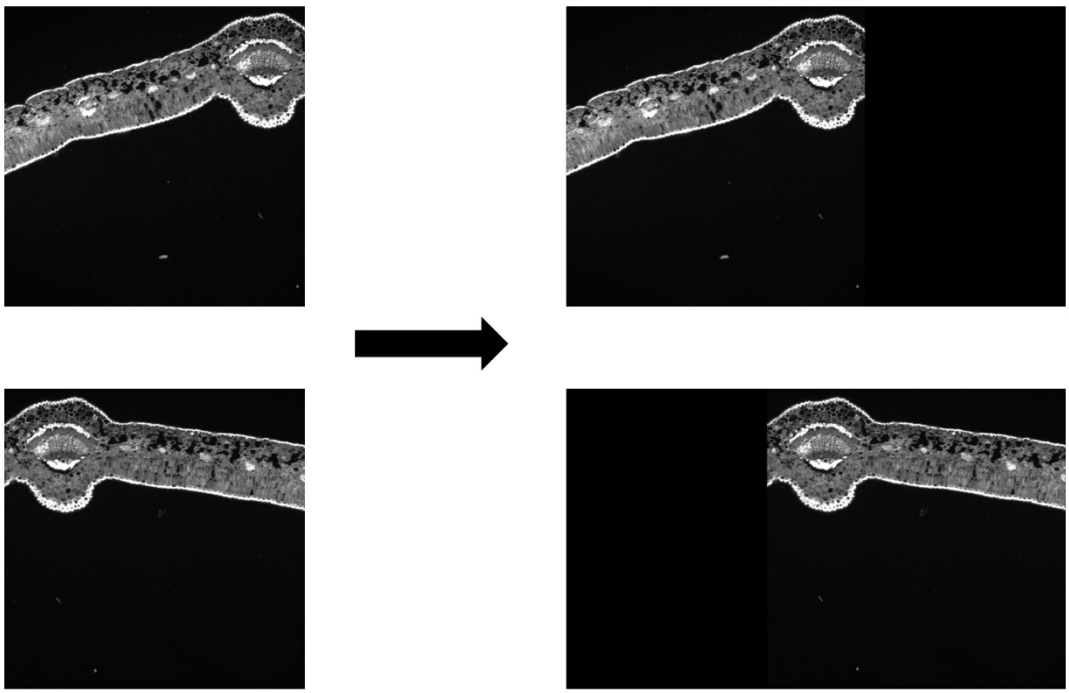

Figura 6.12: Immagini allineate senza perdita di pixel

Per affrontare questo problema abbiamo deciso di "preparare" l'immagine target ad ospitare le immagini che verranno allineate sopra di essa in maniera tale da non avere perdita di dati, in particolare l'idea realizzata consiste nell'espandere i bordi dell'immagine target di una dimensione sufficiente per coprire le dimensioni delle altre immagini una volta allineate. Successivamente le immagini che verranno allineate sull'immagine target avranno una dimensione finale pari a quella del target stesso.

Prima di effettuare l'allineamento viene eseguito un *Preprocess* dell'immagine target. Per ogni immagine da allineare sull'immagine target si eseguono i seguenti passaggi:

1. Viene calcolata la matrice di trasformazione per allineare l'immagine corrente all'immagine target, utilizzando l'algoritmo di allineamento selezionato e i punti corrispondenti delle due immagini.
2. Viene utilizzata la matrice di trasformazione calcolata per trovare le nuove dimensioni dell'immagine target, applicando la traformazione ai vertici del target
3. Viene creata un'immagine completamente nera delle dimensioni totali calcolate e dello stesso tipo dell'immagine target, successivamente l'immagine target originale viene copiata nell'area corrispondente dell'immagine allineata.
4. I Corners dell'immagine target vengono aggiornati in modo tale da non perdere la corrispondenza dei punti trovati in precedenza.

Ripetendo il procedimento su tutte le immagini da allineare il risultato che otteniamo è una nuova immagine target con dimensioni sufficientemente grandi per contenere completamente tutte le immagini da allineare senza perdere alcun pixel.

7 Rendering di Big Data images

Durante lo sviluppo del plugin è stato necessario affrontare il problema della gestione di immagini di grandi dimensioni derivanti ad esempio da scannerizzazioni di interi provini istologici. L'analisi di tali immagini è essenziale per la comprensione delle strutture cellulari, l'identificazione di patologie e lo studio dei processi biologici. L'elaborazione e l'analisi di queste immagini possono presentare diverse sfide. Le immagini istologiche sono spesso caratterizzate da una risoluzione elevata e contengono un gran numero di dettagli, il che comporta una notevole quantità di dati da elaborare. Si è deciso di scartare sin da subito l'opzione di utilizzare una versione di minore qualità rispetto all'immagine caricata, in quanto questo risulterebbe un grosso limite per gli utilizzatori che producono immagini di alta qualità e utilizzando il plugin la perderebbero. Nelle seguenti sezioni viene descritta la soluzione attualmente proposta basata su una famosa libreria chiamata *OpenCV*.

7.1 OpenCV

OpenCV (Open Source Computer Vision Library) è una libreria software ampiamente utilizzata per la gestione e la manipolazione delle immagini in applicazioni di computer vision. È una risorsa molto utile per il processo di allineamento delle immagini in un software dedicato a tale scopo come *DS4H-ImageAlignment*. *OpenCV* fornisce una vasta gamma di funzioni e algoritmi ottimizzati per la trasformazione delle immagini, inclusi metodi per la rotazione, lo scalamento, la traslazione e l'omografia. Questi algoritmi consentono di modificare la posizione, l'orientamento e le dimensioni delle immagini in modo preciso e controllato. *OpenCV* mette a disposizione alcune funzionalità per registrare e allineare le immagini in base a punti di riferimento comuni o caratteristiche specifiche. Ciò consente di correggere distorsioni, adattare le immagini a una prospettiva. Inoltre, *OpenCV* offre anche funzionalità avanzate per il rilevamento e il tracciamento degli oggetti nelle immagini, l'estrazione di punti di interesse, il riconoscimento di pattern e molto altro. Grazie alla sua natura open source, *OpenCV* offre una flessibilità notevole nel suo utilizzo e permette di personalizzare e adattare gli algoritmi in base alle esigenze specifiche del software di allineamento [21].

7.1.1 Gestione immagini tramite *"OpenCV.Mat"*

In *OpenCV*, *Mat* è una struttura di dati fondamentale utilizzata per rappresentare e gestire le immagini. È l'abbreviazione di "matrix" (matrice), che riflette la natura matriciale delle immagini.

La classe *Mat* è progettata per memorizzare i dati dell'immagine in modo efficiente, consentendo l'accesso e la manipolazione dei singoli pixel. È una struttura versatile che può rappresentare immagini in scala di grigi, a colori o con canali multipli. Una *Mat* può essere vista come una matrice bidimensionale di pixel, in cui ogni elemento corrisponde a un pixel dell'immagine. La dimensione della matrice corrisponde alle dimensioni dell'immagine, mentre il tipo di dato specifica il formato dei pixel (ad esempio, unsigned integer, floating point, etc.). *OpenCV* fornisce un'ampia gamma di funzioni e metodi per lavorare con oggetti *Mat*.

7.2 Gestione immagini tramite *"ij.ImagePlus"*

In *ImageJ* le immagini vengono rappresentate come istanze della classe *ij.ImagePlus*. Questa classe fornisce un'astrazione per un insieme di slice di immagine *(ij.ImageStack)*. Ogni slice rappresenta un'immagine bidimensionale e il tipo di dati dell'immagine dipende dalla classe *ij.process.ImageProcessor* utilizzata.

Ci sono diverse classi *ImageProcessor* disponibili, tra cui [22]:

- *ij.process.ByteProcessor* per immagini in scala di grigi a 8 bit,
- *ij.process.ShortProcessor* per immagini a scala di grigi a 16 bit,
- *ij.process.FloatProcessor* per immagini in scala di grigi a precisione singola 16 bit,
- *ij.process.ColorProcessor* per immagini a colori a 24 bit.

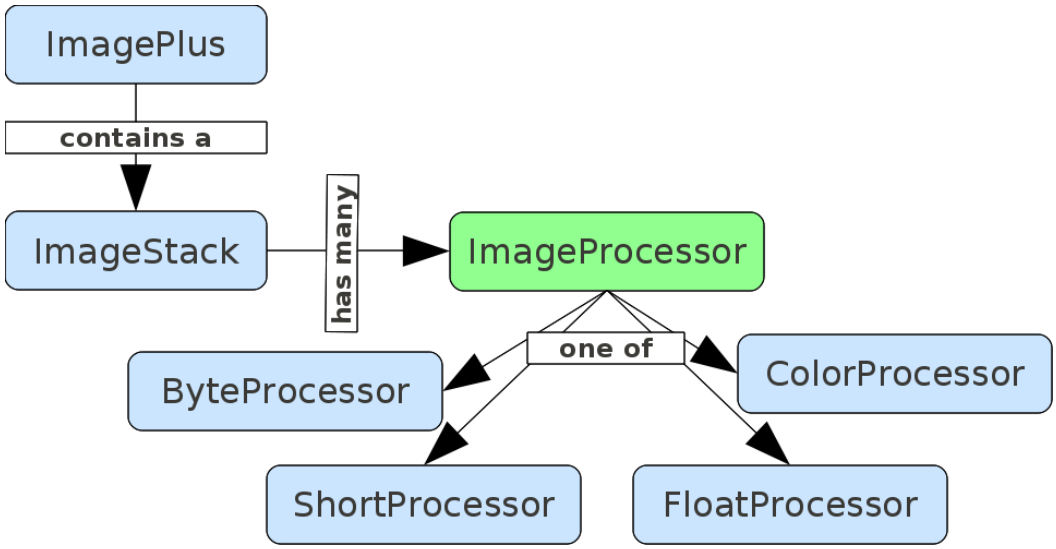

Figura 7.1: Struttura per gestione delle immagini in ImageJ

7.3 Gestione immagini in DS4H-ImageAlignment

Per il software sviluppato, era necessario sfruttare entrambi i vantaggi, l'ottima gestione delle immagini offerta da *ImagePlus* e degli strumenti comodi forniti dalla libreria *OpenCV*. Di conseguenza, sono state create due classi apposite per consentire una conversione efficiente tra le due rappresentazioni delle immagini.

```
private static FloatProcessor makeFloatProcessor(Mat matrix, final int width,
final int height){
    final FloatProcessor floatProcessor = new FloatProcessor(width, height);
    if(matrix.channels() > 1) {
        Imgproc.cvtColor(matrix, matrix, Imgproc.COLOR_BGR2GRAY);
    }
    matrix.convertTo(matrix, CvType.CV_32FC1);
    matrix.get(0,0, (float[])floatProcessor.getPixels());
    matrix.release();
    matrix = null;
    return floatProcessor;
}
```

Figura 7.2: Metodo per la conversione da OpenCV.Mat a ij.FloatProcessor

```
private static Mat toMat(final FloatProcessor fp){
    final Mat matrix = new Mat(fp.getHeight(), fp.getWidth(),
CvType.CV_32FC1);
    matrix.put(0,0, (float[]) fp.getPixels());
    return  matrix;
}
```

Figura 7.3: Metodo per la conversione da ij.FloatProcessor a OpenCV.Mat

Per affrontare questa sfida, abbiamo adottato l'approccio di sfruttare sia le funzionalità di *OpenCV* sia la gestione ottimizzata offerta da *ImagePlus*. In particolare, abbiamo utilizzato la classe "*ImagePoints*" come estensione della classe base "*ImagePlus*" fornita da *ImageJ/Fiji*. Questa classe è stata appositamente progettata per consentire una gestione efficiente delle immagini, fornendo funzionalità specifiche come l'aggiunta di punti di riferimento (Corners) e introducendo metodi e attributi aggiuntivi per semplificarne la gestione. Utilizzando "*ImagePoints*" come rappresentazione delle immagini nel nostro plugin, siamo stati in grado di beneficiare sia delle potenti funzioni di elaborazione delle immagini offerte da *OpenCV*, sia delle caratteristiche ottimizzate per la gestione delle immagini fornite da *ImageJ/Fiji*.

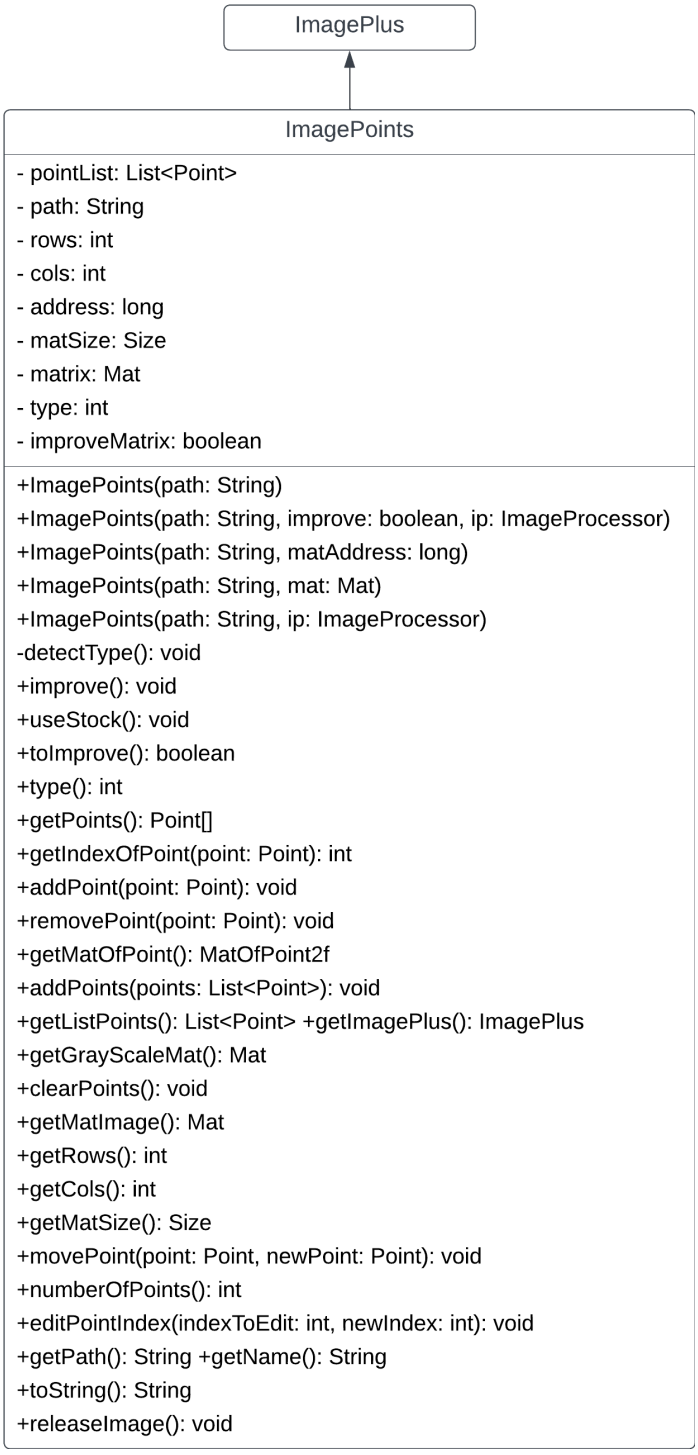

Figura 7.4: Struttura ImagePoints

8 Risultati sperimentali

8.1 Scenario di utilizzo: allineamento manuale dopo automatico

Un tipico scenario facilmente riscontrato durante l'utilizzo del tool da parte dei Medici e Biologi nostri referenti prevede i seguenti step:

1. Tentativo di allineare alcune immagini con un algoritmo di allineamento automatico per avere un vantaggio in termini di tempo (ricordiamo che con l'allineamento manuale sarebbe necessario inserire una serie di Corner Points per ogni immagine da allineare).
2. Dopo l'allineamento automatico una o più immagini potrebbero non essere allineate correttamente *(vedi immagine DIC, Figura 8.1)*.
3. Il modo migliore per proseguire ad allineare le immagini non correttamente trasformate è di salvare lo stack attuale escludendo le immagini non allineate correttamente.
4. Effettuare il reuse dell'immagine target o di una qualsiasi immagine correttamente allineata.
5. Caricare nuovamente le immagini non correttamente allineate attraverso il metodo automatico.
6. Inserire dei Corner Points sulle immagini e procedere con l'allineamento manuale sullo stack di immagini.
7. Salvare il nuovo stack con tutte le immagini tutte correttamente allineate.

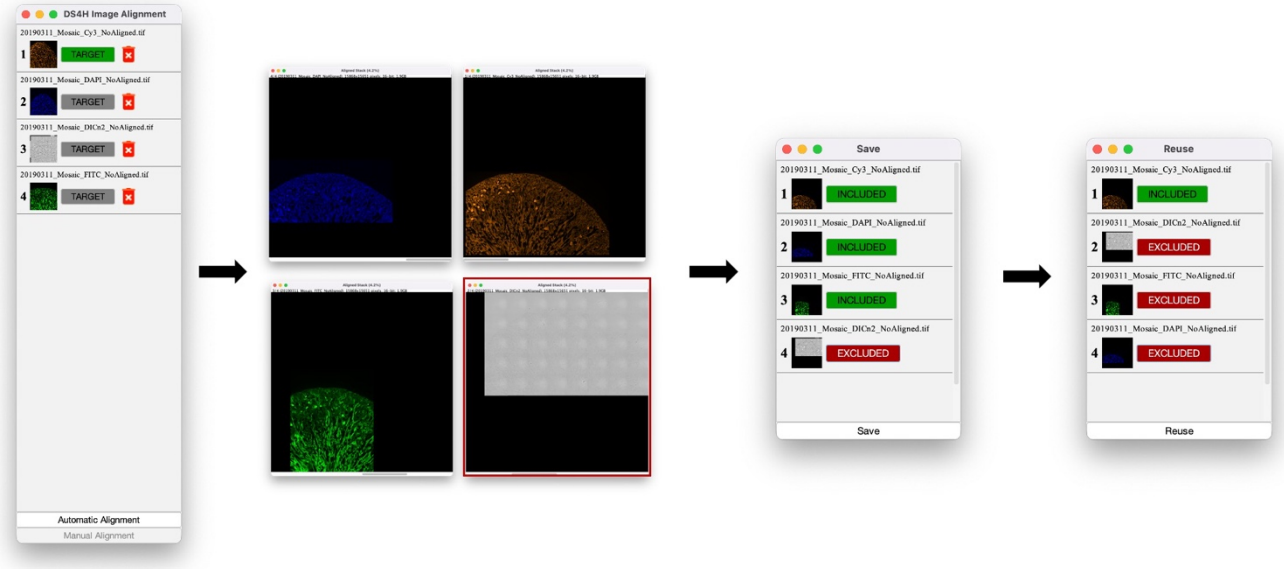

Figura 8.1: Allineamento errato > Save > Reuse

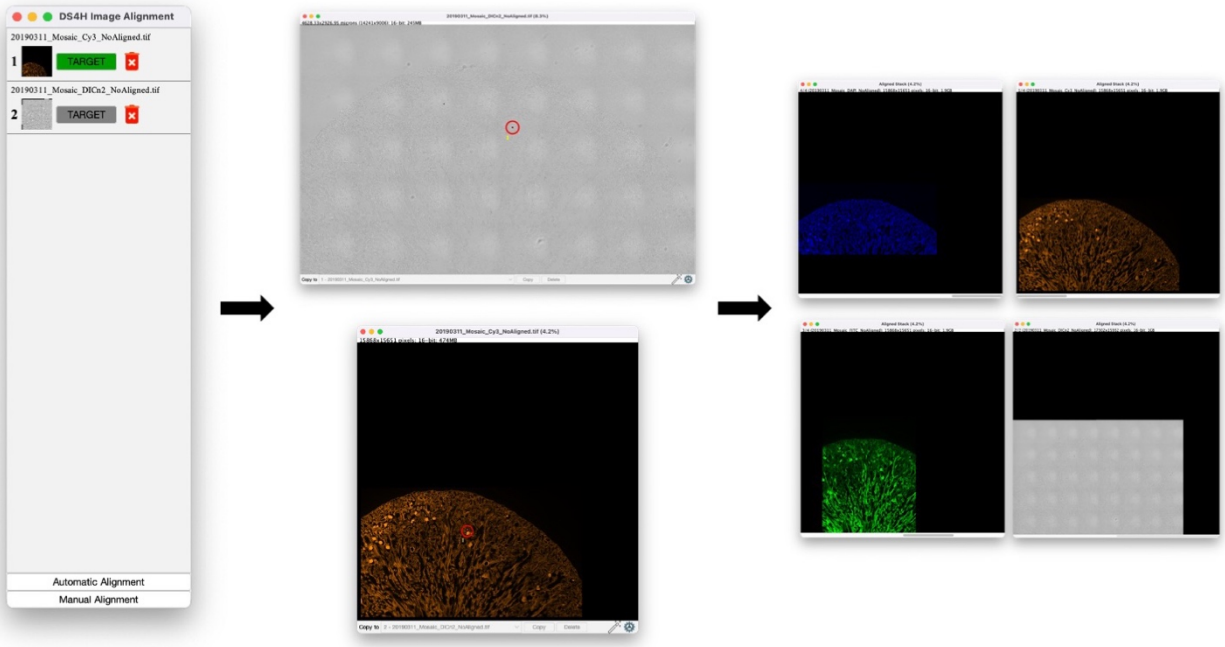

Figura 8.2: Allineamento manuale a seguito di un errato allineamento automatico

Questo procedimento ci permette di avere un grosso vantaggio in termini di tempo, effettuare un allineamento completamente manuale richiederebbe di inserire un numero di corner sufficiente per ogni immagine del nostro stack di allineamento. Invece, in questo modo possiamo prima di tutto effettuare un allineamento automatico in maniera rapida, nella speranza che la maggior parte delle immagini vengano allineate nel modo corretto, e solo successivamente se ci dovessero essere immagini non correttamente allineate possiamo procedere nell'aggiungerle attraverso un allineamento manuale.

8.2 Test con dataset di prova

Il plugin è stato testato attraverso due dataset di prova, cercando di allineare automaticamente immagini acquisite con differenti marcatori fluorescenti per nucleo (i.e. DAPI), membrana (i.e. Cy3), citoplasma (i.e. FITC) ed anche non fluorescenti (ad esempio attraverso la tecnica conosciuta in microscopia con il nome Differential Interference Contrast - DIC), e diversi fattori di sovrapposizione (imposto come spostamento approssimativo tra due immagini seguenti acquisite manualmente al microscopio)

- XY1: shift 0%
- XY2: shift 25%
- XY3: shift 50%
- XY4: shift 75%

I dataset sono stati registrati utilizzando come algoritmo di allineamento automatico quello denominato con *SURF*. Nonostante in molti casi si riesca ad effettuare un allineamento ottimo anche con shift fino al 75%, non sempre è possibile ottenere un allineamento ottimale in maniera automatica. Tuttavia, usare gli algoritmi automatici anche solo per registrare un subset di imamgini porta ad avere un guadagno in termini di tempo non indifferente.

I seguenti dati si riferiscono ad un test di allineamento su un primo dataset di esempio, sezione di rene di topo, con immagini aventi diverse percentuali di spostamento e differenti marcatori fluorescenti, in particolare: Cy3, DIC, DAPI, FITC. Per ogni spostamento ed ogni marcatore è stato valutato l'allineamento:

- Y → allineamento ben riuscito
- N → allineamento non riuscito correttamente

		XY1				XY2				XY3				XY4			
		Cy3	DIC	DAPI	FITC	Cy3	DIC	DAPI	FITC	Cy3	DIC	DAPI	FITC	Cy3	DIC	DAPI	FITC
XY1	Cy3	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	N	N	N	N	N	N	N
	DAPI	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	Y	Y

Figura 8.3: Risultati test allineamento automatico con SURF [14] DataSet 1

- Per le immagini del canale Cy3:
    - Numero di allineamenti corretti: 12/16 (75%)
    - Numero di allineamenti errati: 4/16 (25%)
- Per le immagini della modalità DIC:
    - Numero di allineamenti corretti: 1/16 (6,5%)
    - Numero di allineamenti errati: 15/16 (93,5%)
- Per le immagini del canale DAPI:
    - Numero di allineamenti corretti: 8/16 (50%)
    - Numero di allineamenti errati: 8/16 (50%)
- Per le immagini del canale FITC:
    - Numero di allineamenti corretti: 12/16 (75%)
    - Numero di allineamenti errati:4/16 (25%)

L'allineamento automatico è stato poi visivamente valutato su un secondo dataset di esempio, un provino istologico contenente una scaglia pelosa di Elaeagnus commutata argentea, con immagini aventi diverse percentuali di spostamento e differenti marcatori fluorescenti, in particolare: Cy3, DIC, CY5, FITC.

		XY1				XY2				XY3				XY4			
		Cy3	DIC	CY5	FITC	Cy3	DIC	CY5	FITC	Cy3	DIC	CY5	FITC	Cy3	DIC	CY5	FITC
XY1	Cy3	Y	N	N	Y	Y	Y	N	N	Y	N	N	N	N	N	N	N
	DIC	N	Y	N	Y	N	Y	N	N	N	Y	N	N	N	Y	N	N
	CY5	N	N	Y	N	N	N	Y	N	N	N	Y	N	N	N	Y	N
	FITC	Y	Y	N	Y	Y	N	N	Y	N	Y	N	Y	N	Y	N	Y

Figura 8.4: Tabella di test allineamento automatico tramite SURF [17] DataSet 2

- Per le immagini del canale Cy3:
    - Numero di allineamenti corretti: 5/16 (31%)
    - Numero di allineamenti errati: 11/16 (69%)
- Per le immagini della modalità DIC:
    - Numero di allineamenti corretti: 5/16 (31%)
    - Numero di allineamenti errati: 11/16 (69%)
- Per le immagini del canale CY5:

    - Numero di allineamenti corretti: 4/16 (25%)
    - Numero di allineamenti errati: 12/16 (75%)
- Per le immagini del canale FITC:
    - Numero di allineamenti corretti: 9/16 (56%)
    - Numero di allineamenti errati: 7/16 (44%)

Valutando complessivamente i risultati ottenuti, possiamo dire che l'algoritmo automatico testato porta ad allineare correttamente circa il 40% delle immagini. Notiamo come le immagini DIC risultano spesso difficili da allineare attraverso un allineamento automatico questo è dato dal fatto che le immagini DIC spesso presentano un contrasto complesso e variazioni di intensità molto sottili. Questo rende difficile per gli algoritmi di allineamento automatico riconoscere e confrontare in modo accurato le caratteristiche tra le immagini, influenzando negativamente i risultati dell'allineamento e rendendo necessaria una procedura di allineamento semi-automatico.

9 Conclusioni e sviluppi futuri

In questa tesi, è stato sviluppato il tool "*DS4H Image Alignment*", un plug-in di facile utilizzo per l'allineamento rapido ed intuitivo di immagini acquisite con microscopi a campo largo. Il progetto è stato realizzato all'interno del gruppo di ricerca "*Data Science for Health*" (DS4H), composto da ricercatori e professori dell'Università di Bologna (UniBo) e dell'IRCCS Istituto Romagnolo dei Tumori Dino Amadori (IRST) di Meldola (FC).

L'obiettivo principale del progetto è stato quello di colmare la mancanza di una procedura standard per l'allineamento delle immagini istologiche acquisite con diverse tecniche e colorazioni. Attualmente, l'allineamento manuale è un processo dispendioso in termini di tempo e soggetto a possibili errori umani. Con l'introduzione di *DS4H Image Alignment*, si è cercato di fornire una soluzione tecnica e automatica per l'allineamento delle immagini mantenendo comunque la possibilità di effettuare allineamenti anche manuali in maniera intuitiva, consentendo una migliore valutazione dei provini istologici e facilitando lo studio di colocalizzazione dei segnali intracellulari.

Attraverso la collaborazione con medici e biologi dell'IRST, sono state identificate esigenze e richieste degli operatori del settore, che hanno contribuito alla definizione dei requisiti finali del plug-in. I risultati ottenuti attraverso l'implementazione di *DS4H Image Alignment* sono promettenti: Il plug-in ha dimostrato di consentire un allineamento rapido e affidabile delle immagini, facilitando l'analisi dei provini istologici e consentendo una migliore valutazione dei segnali di colocalizzazione. L'utilizzo del modulo di gestione smart corners e del rendering delle immagini ha contribuito a ottimizzare ulteriormente il processo di allineamento.

DS4H Image Alignment presenta diversi vantaggi rispetto ai suoi concorrenti. Il plugin è ora in grado di **gestire immagini di grandi dimensioni** e riesce ad allinearle in maniera efficiente, offrendo una soluzione pratica e affidabile per l'allineamento di dati di dimensioni complesse. Questo è un aspetto fondamentale considerando la crescente quantità di dati generati dalle moderne tecniche di imaging.

Il plugin offre **un'interfaccia utente molto intuitiva e di facile utilizzo**, intuitiva e ben progettata, grazie ad essa gli operatori possono eseguire l'allineamento delle immagini senza la necessità di competenze tecniche avanzate. Ciò rende il plugin accessibile a un'ampia gamma di utenti, inclusi

medici, biologi e ricercatori che potrebbero non essere esperti in informatica o in elaborazione delle immagini.

Un ulteriore vantaggio del plugin è la facilità di **gestione e riutilizzo delle immagini**. In caso di errori o necessità di ulteriori allineamenti, il plugin consente una gestione semplice delle immagini, permettendo di **ripetere il processo di allineamento con facilità**. Questo è particolarmente vantaggioso in un contesto di ricerca o clinico in cui è necessario esplorare diverse configurazioni e valutare diverse opzioni di allineamento.

Personalmente, durante lo sviluppo del plugin mi sono occupato di strutturare il progetto secondo le linee guida del pattern *MVC*, dell'implementazione e la gestione dei corner point per trovare le corrispondenze tra le immagini, dell'implementazione dell'algoritmo Traslativo, Proiettivo ed Affine per gli allineamenti manuali, la gestione ottimizzata e il riutilizzo delle immagini per effettuare allineamenti successivi ed infine ho dedicato del tempo alla validazione del tool usando alcuni dataset di test.

La versione attuale del plugin è stata menzionata nei seguenti contributi a conferenza:

- ***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. Poster Presentation
- ***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.

Infine, per rendere più popolare il tool alla comunità scientifica e raggiungere un numero più consistente di possibili utilizzatori, stiamo ora procedendo nella preparazione di un articolo scientifico da essere sottomesso a rivista con Impact Factor.

Bibliografia

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

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

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

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

[5] Kevin W. Eliceiri, Michael R. Berthold, Ilya G. Goldberg, Luis Iba ́n ̃ez, B. S. Manju- nath, Maryann E. Martone, Robert F. Murphy, Hanchuan Peng, Anne L. Plant, Badri- nath Roysam, Nico Stuurman, Jason R. Swedlow, Pavel Tomancak, and Anne E. Car- penter. Biological imaging software tools. *Nature methods*, 9(7):697–710, Jun 2012. 22743775[pmid].

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

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

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

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

[10] Johannes Schindelin, Ignacio Arganda-Carreras, Erwin Frise, Verena Kaynig, Mark Lon- gair, Tobias Pietzsch, Stephan Preibisch, Curtis Rueden, Stephan Saalfeld, Benjamin Sch- mid, Jean-Yves Tinevez, Daniel James White, Volker Hartenstein, Kevin Eliceiri, Pa- vel Tomancak, and Albert Cardona. Fiji: an open-source platform for biological-image analysis. *Nature Methods*, 9:676 EP –, Jun 2012. Perspective.

[11] ”Developing an ImageJ Plugin” URL: https://imagej.net/develop/ij1-plugins,

[12] Intellij idea. https://www.jetbrains.com/idea/.

[13] Frederic P Miller, Agnes F Vandome, and John McBrewster. Apache maven. 2010. URL: https://maven.apache.org/,

[14] 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. URL: https://www.sciencedirect.com/science/article/pii/S1077314207001555,

[15] Sift URL: https://it.wikipedia.org/wiki/Scale-invariant_feature_transform

[16] Arthur Ardeshir Goshtasby. *2-D and 3-D image registration: for medical, remote sensing, and industrial applications*. John Wiley & Sons, 2005.

[17] Jacques Feldmar and Nicholas Ayache. Rigid, affine and locally affine registration of free-form surfaces. *International journal of computer vision*, 18(2):99–119, 1996,

[18] Retta stimata con RANSAC URL: https://it.wikipedia.org/wiki/RANSAC

[19] Chum, O., Matas, J., & Kittler, J. (2005). Locally optimized RANSAC. In Computer Vision and Pattern Recognition, 2005. CVPR 2005. IEEE Computer Society Conference on (Vol. 1, pp. 775-781). IEEE,

[20] Scott Schaefer, Travis McPhail, and Joe Warren. Image deformation using moving least squares. *ACM Trans. Graph.*, 25:533–540, 07 2006,

[21] OpenCV. "OpenCV: Open Source Computer Vision Library." Versione 4.5.3. URL: https://opencv.org/,

[22] ImagePlus. “ImagePlus: The hierarchy of the classes representing an image” URL: https://imagej.net/develop/ij1-plugins,

[23] David Levin. The approximation power of moving least-squares. *Mathematics computation*, 67(224):1517–1531, 1998,

Ringraziamenti

Desidero esprimere un sentito ringraziamento alla Prof. Antonella Carbonaro, mia Relatrice, per avermi offerto l'opportunità di partecipare a questo affascinante progetto di ricerca. Sono grato per la possibilità di dimostrare le mie competenze e contribuire a un ambito così stimolante come quello del *Data Science for Health*.

Desidero inoltre ringraziare i Professori Giovanni Martinelli e Gastone Castellani per aver reso possibile il progetto e per aver concesso l'utilizzo delle preziose immagini reali acquisite presso l'Istituto Romagnolo per lo Studio dei Tumori "Dino Amadori" IRCCS di Meldola. La disponibilità di queste immagini è stata fondamentale per lo sviluppo e la validazione del plug-in *DS4H Image Alignment.*

Un ringraziamento speciale va al Prof.Filippo Piccinini, che ha svolto il ruolo di correlatore in questa ricerca. La sua presenza costante, i consigli preziosi e il supporto sono stati fondamentali per ottenere ottimi risultati. La sua conoscenza nel campo della microscopia e dell'oncologia è stata condivisa in modo chiaro ed esaustivo, semplificando argomenti complessi. La sua gentilezza e la sua ecletticità hanno reso questa esperienza ancora più gratificante.

Infine, desidero ringraziare la famiglia e gli amici che mi hanno sostenuto durante tutto il percorso di realizzazione di questa tesi. Il loro sostegno e incoraggiamento sono stati fondamentali per affrontare le sfide e perseverare nel raggiungimento degli obiettivi.Ringrazio tutti coloro che hanno contribuito in modo significativo a questa tesi, rendendo possibile il suo completamento e arricchendo l'esperienza di ricerca.

Desidero esprimere la mia sincera gratitudine ai miei amici più stretti, in particolare a Elia, Giacomo, Giulia e Alberto, ringrazio anche gli amici Matteo, Nicola, Samuele e Filippo, con i quali ormai da tanto tempo condividiamo un pezzo di strada importante. La vostra presenza durante tutto il percorso universitario è stata fondamentale. Il supporto reciproco, i momenti di svago e le risate condivise hanno reso questa esperienza ancora più piacevole e significativa.

Un ringraziamento speciale va alla mia famiglia, che è stata il mio fondamento e supporto costante lungo tutto questo percorso.