Politecnico di Torino

POLITECNICO DI TORINO

Dipartimento di Ingegneria Informatica, Cinema e Meccatronica

Corso di Laurea in Ingegneria Informatica

Tesi di Laurea Magistrale

Progettazione e Sviluppo di un Sistema Software Integrato per l'Elaborazione Avanzata e l'Analisi Morfologica di Immagini Cometarie

Relatori
prof. Bartolomeo Montrucchio
dr. Antonio Costantino Marceddu

Candidato
Fabio Genovese

Marzo 2026

Abstract

Il presente lavoro di tesi descrive le fasi di progettazione e sviluppo di un applicativo software multipiattaforma dedicato all'analisi morfologica e alla gestione di immagini astronomiche di comete. Tale progetto è stato portato a termine attraverso un approccio iterativo, realizzato come collaborazione tra il Politecnico di Torino ed astrofili ricercatori affiliati all'Osservatorio Astrofisico di Asiago. Ciò ha consentito un confronto costante con gli esperti di dominio, permettendo di validare le scelte architetturali e di calibrare gli algoritmi su dati reali. Questo processo ha portato alla realizzazione di Kometra, uno strumento software che risponde efficacemente alle esigenze operative concrete della ricerca astrofisica.

Kometra è stato realizzato utilizzando il linguaggio C# e il framework Avalonia UI basato su piattaforma .NET. Tale scelta architetturale ha permesso di garantire l'esecuzione nativa e la piena portabilità sui principali sistemi operativi, rispondendo all'esigenza di operare in ambienti di ricerca eterogenei. Il cuore computazionale del sistema si basa sull'integrazione della libreria OpenCV, le cui primitive ad alte prestazioni per il calcolo matriciale e la computer vision sono state sfruttate per l'ottimizzazione degli algoritmi di elaborazione.

Il software è progettato per gestire correttamente dati astronomici in formato Flexible Image Transport System (FITS), integrando in modo nativo la compressione e decompressione standard dei file. La logica di gestione degli header garantisce la conservazione dei metadati originali e permette sia la modifica manuale dei parametri, sia l'aggiornamento automatico delle chiavi a seguito di trasformazioni geometriche o fotometriche, assicurando la piena conformità agli standard astronomici e la riproducibilità delle analisi.

Il flusso di elaborazione è stato ottimizzato per lo studio di oggetti in movimento non siderale: l'applicativo supporta l'integrazione dei dati orbitali provenienti dal sistema Horizons del Jet Propulsion Laboratory (JPL) e delle coordinate astrometriche definite secondo lo standard World Coordinate System (WCS), permettendo una gestione ancora più rigorosa del moto relativo rispetto al campo stellare. Su questa base, sono state sviluppate e comparate diverse strategie per il rilevamento robusto dell'optocentro corrispondente al nucleo cometario, il mascheramento automatico delle sorgenti stellari di fondo, il successivo allineamento e lo stacking delle acquisizioni, passaggi fondamentali per incrementare il rapporto Signal-to-Noise Ratio (SNR) della chioma cometaria.

Inoltre, per permettere un'analisi approfondita delle caratteristiche morfologiche, il software integra diversi filtri digitali provenienti da ambiti differenti. Da un lato vengono impiegate metodologie astronomiche standard, essenziali per la modellazione della chioma e l'isolamento delle strutture cometarie: tra queste spiccano filtri basati sui gradienti radiali e rotazionali come il Larson-Sekanina, e le tecniche basate su modelli sintetici della chioma. Dall'altro, sono stati integrati algoritmi di computer vision, come l'equalizzazione Contrast Limited Adaptive Histogram Equalization (CLAHE) e, a titolo sperimentale, alcune tecniche derivate dall'imaging biomedico, come il filtro di Frangi.

A completamento del set di strumenti, l'applicativo offre funzionalità di esportazione di immagini e video, consentendo di sintetizzare le sequenze di immagini elaborate in animazioni fluide.

Indice

Elenco delle figure

Elenco delle tabelle

Acronimi

A

ADC Analog-to-Digital Converter 17
ADU Analog-to-Digital Unit viii, 13, 17, 95
API Application Programming Interface 51, 52
AR Ascensione Retta 56
ASCII American Standard Code for Information Interchange 15, 16, 20
ASTAP Astro Stacking and Astrometric Plate solver iv, vi, 51, 52, 57, 58
AVI Audio Video Interleave 28

C

CCD Charge-Coupled Device 13, 17, 19, 21, 34
CLAHE Contrast Limited Adaptive Histogram Equalization ii, viii, 81, 87
CMOS Complementary Metal-Oxide Semiconductor 13, 16, 17, 34
CPU Central Processing Unit 17, 67

D

DEC Declinazione 56
DI Dependency Injection 9, 10
DSO Deep Sky Object 63

E

EOF End Of File 19
ESA European Space Agency 30

F

FFT Fast Fourier Transform (Trasformata di Fourier Veloce) vi, 54
FITS Flexible Image Transport System ii, vi, 1, 2, 4, 7, 9–11, 13–17, 19, 21, 22, 24, 25, 27, 35, 50, 52, 55–57, 68, 69, 72, 79, 81, 86, 87, 91–93, 96
FOV Field of View 34
FWHM Full Width at Half Maximum 65

G

GPU Graphics Processing Unit 97
GZIP GNU Zip iii, 21, 22, 27

H

HDU Header Data Unit 15, 19, 20, 24, 57
HEVC High Efficiency Video Coding 28
HTML HyperText Markup Language 7

I

I/O Input/Output 9, 13
IAU International Astronomical Union 14, 17
IAU-FWG IAU FITS Working Group 14
IoC Inversion of Control vi, 9

J

JPEG Joint Photographic Experts Group 13, 16, 27
JPL Jet Propulsion Laboratory ii, iv, 50, 52, 53, 58, 59

L

LoG Laplacian of Gaussian 65
LOH Large Object Heap 55
LSN Local Spatial Normalization vii, 78, 79

M

MCM Median Coma Model vii, 73–76, 87
MEF Multi-Extension FITS iii, vi, 19, 24, 27
MJPG Motion JPEG 28
MKV Matroska Video 28
MP4 MPEG-4 Part 14 28
MVVM Model-View-ViewModel 8, 9

N

NaN Not-a-Number 63, 66, 67, 79, 92
NASA National Aeronautics and Space Administration 22, 52, 56
NRAO National Radio Astronomy Observatory 14

P

PNG Portable Network Graphics 13, 16, 27
POCO Plain Old CLR Object 8
PSF Point Spread Function 33, 34, 52

R

R&D Research and Development 66
RAM Random Access Memory 7, 14, 16, 17, 57, 67, 69, 78, 80, 87
ROI Region of Interest 36
RVSF Radial Variance Spatial Filter vii, 77, 78
RWM Radial Weighted Model vii, 74, 75, 87

S

SBP Space-Bandwidth Product 34
SIP Simple Imaging Polynomial 57
SNN Symmetric Nearest Neighbours viii, 82, 83
SNR Signal-to-Noise Ratio ii, iv, 1, 35

T

TPV Tangent Plane and Polynomial vi, 56, 57

U

UA Unità Astronomica 29
UI User Interface 5–8, 24
UV Ultravioletto 33

V

VLA Very Large Array 14

W

WCS World Coordinate System ii, iv, vi, 14, 26, 50–53, 55–59
WPF Windows Presentation Foundation 7

X

XAML eXtensible Application Markup Language 7, 8

Y

Y2K Year 2000 14

Capitolo 1

Introduzione

L'elaborazione computazionale di immagini astronomiche impone requisiti architetturali e algoritmici profondamente diversi da quelli della fotografia digitale convenzionale. Mentre i formati grafici classici applicano compressioni e curve di *tone mapping* non lineari per adattare l'immagine alla percezione visiva umana, il software astrofisico deve trattare il fotogramma come un set di dati scientifici crudi. Per consentire misurazioni fotometriche e morfologiche accurate, è imperativo che l'infrastruttura di elaborazione mantenga in memoria matrici ad altissima precisione, preservando la fedeltà del segnale originario e l'intera gamma dinamica catturata dalla strumentazione.

All'interno di questo vasto dominio, l'osservazione e l'analisi delle comete presentano una serie di sfide computazionali uniche. Infatti, a differenza delle sorgenti celesti statiche (come galassie o ammassi stellari), le comete sono oggetti complessi e intrinsecamente dinamici. La loro elaborazione software impone di risolvere diverse problematiche, che riflettono parallelamente l'intera *pipeline* analitica:

1. **Gestione dei Formati Scientifici:** L'architettura software deve supportare nativamente standard astronomici specializzati come il formato FITS [1], manipolando enormi matrici di dati anche in virgola mobile e gestendo complesse architetture di metadati e compressione *lossless* [2], al fine di evitare qualsiasi degradazione fotometrica preliminare.

2. **Ricerca del Nucleo e Allineamento Dinamico:** Tracciare il movimento di un oggetto diffuso è un problema non banale nell'ambito della *Computer Vision*. La cometa presenta un gradiente sfumato e asimmetrico che rende estremamente difficile la misurazione del suo centro geometrico. Una volta individuato l'optocentro con precisione sub-pixel, il sistema deve traslare e ruotare i fotogrammi per inseguire dinamicamente il bersaglio lungo la sua traiettoria.

3. **Separazione del Campo Stellare e Stacking:** L'allineamento sulle coordinate del nucleo cometario genera il fisiologico "strisciamento" (*trailing*) delle stelle presenti nello sfondo nei fotogrammi. Prima di procedere alla fusione del segnale (*stacking*), necessaria per massimizzare il rapporto SNR, è fondamentale isolare e mascherare le interferenze stellari, evitando che le tracce stellari inquinino la chioma.

4. **Estrazione tramite Filtri Morfologici:** L'intensità luminosa emessa dalla cometa decade vertiginosamente allontanandosi dal nucleo, creando un velo luminoso preponderante che sommerge le micro-strutture di reale interesse scientifico (come i getti di polvere o i gusci di espansione). Per questa ragione, spesso i tradizionali operatori di contrasto si rivelano inefficaci, rendendo necessaria l'applicazione di filtri spaziali e tensoriali avanzati, capaci di sopprimere il gradiente isotropo e isolare le anisotropie senza saturare irrimediabilmente l'immagine.

Nonostante il proliferare di applicativi dedicati all'elaborazione astrofotografica, il panorama degli strumenti a disposizione della comunità per lo studio specifico della morfologia cometaria

risulta ancora oggi carente e frammentato. La maggior parte dei software *open-source* attualmente in uso si presenta infatti come uno strumento generalista, concepito per processare un'ampia varietà di immagini astrofisiche. Di conseguenza, questi applicativi offrono strumenti di indagine ampi ma superficiali, mancando di funzionalità verticali e ottimizzate per le peculiarità dinamiche e fotometriche delle comete appena descritte. Inoltre, questi applicativi si basano tipicamente su interfacce utente obsolete e poco reattive, non sfruttano appieno le moderne architetture hardware e mancano di un *feedback* visivo interattivo. Tipicamente l'operatore si trova costretto a concatenare software eterogenei per coprire tutte le fasi dell'analisi: dall'allineamento alla rimozione delle stelle (*Starless*), fino all'applicazione dei filtri spaziali. Tali passaggi continui di esportazione e importazione non solo dilatano i tempi operativi, ma mettono a rischio l'integrità dei metadati e la precisione del dato originario.

Per rispondere a queste criticità e fornire un ambiente di lavoro unificato, il presente lavoro di tesi illustra la progettazione e lo sviluppo di **Kometra** [3]: un sistema software integrato, multipiattaforma e ad alte prestazioni, concepito esclusivamente per la gestione, l'elaborazione astrometrica e l'analisi morfologica avanzata di immagini cometarie.

La realizzazione di questo applicativo ha seguito un approccio iterativo, fondato su una stretta collaborazione tra il Politecnico di Torino e alcuni ricercatori affiliati all'Osservatorio Astrofisico di Asiago. Il costante confronto con gli esperti di dominio ha permesso di tradurre problematiche analitiche reali in requisiti software ben definiti, validando le scelte architetturali direttamente su dataset scientifici concreti. Dal punto di vista dell'Ingegneria Informatica, Kometra [3] è stato sviluppato adottando il linguaggio C# e il *framework* .NET, accoppiati alla piattaforma Avalonia UI [4]. Questa combinazione architetturale ha permesso la creazione di un'applicazione con esecuzione nativa su ambienti Windows, macOS e Linux, garantendo uniformità di interfacce e prestazioni. Per sopperire al carico computazionale richiesto dalle manipolazioni matriciali ad alta precisione, il *backend* matematico è stato ancorato alle routine ottimizzate della libreria OpenCV [5], [6], governate da un'attenta gestione della memoria in *multithreading*.

L'innovazione apportata dal progetto si articola su diverse direttrici fondamentali. In primo luogo, Kometra [3] integra un'infrastruttura dati robusta, basata su un motore di *parsing* proprietario per lo standard astronomico FITS e per la *Tiled Image Compression*, assicurando la conservazione dei metadati in ogni fase di lavoro. A ciò si aggiungono avanzate funzionalità di astrometria e *tracking* dinamico. Per affrontare il problema delle interferenze visive, è stata sviluppata una complessa *pipeline* algoritmica volta alla rimozione automatica delle stelle di campo. A questo si unisce un potente motore di analisi morfologica, che fornisce all'astronomo una vasta libreria di filtri spaziali avanzati: dai classici operatori rotazionali come il Larson-Sekanina [7] a modelli statistici adattivi, fino ad approcci mutuati dalla *Computer Vision*. Infine, l'intero sistema è sorretto da un paradigma operativo non distruttivo basato su un'architettura a nodi (*Node Tree*), che permette all'utente di condurre parallelamente test alternativi, validando i risultati in tempo reale senza mai alterare i dati sorgente.

Il presente documento descrive in modo analitico e sequenziale le sfide ingegneristiche affrontate, i modelli matematici adottati e le relative soluzioni implementative, organizzando la trattazione come segue:

Il **Capitolo 2** definisce i requisiti di dominio e l'architettura generale del sistema. Vengono illustrate le motivazioni tecnologiche, i *design pattern* utilizzati e l'implementazione logica dello spazio di lavoro basato sui nodi operativi.

Il **Capitolo 3** esplora nel dettaglio l'infrastruttura di gestione dei dati astronomici. Il focus è posto sull'ingegnerizzazione del *parser* per il formato FITS, sulla gestione della memoria per matrici di grandi dimensioni in virgola mobile e sullo sviluppo dei motori di codifica per la compressione.

Il **Capitolo 4** tratta le funzionalità di pre-elaborazione, calibrazione spaziale e *tracking*. Vengono analizzati gli strumenti sviluppati per la visualizzazione dell'*header*, la risoluzione astrometrica, il rilevamento del centroide cometario e le conseguenti operazioni di allineamento e *stacking* dei fotogrammi.

Il **Capitolo 5** è dedicato al problema del mascheramento del campo stellare. Il capitolo analizza la *pipeline* algoritmica impiegata per identificare, mascherare e rimuovere le stelle, illustrando infine le tecniche per la ricostruzione realistica del segnale mancante del fondo cielo.

Il **Capitolo 6**, infine, si addentra nel cuore analitico dell'applicativo: il motore di filtraggio morfologico. Vengono esposti i fondamenti teorici e l'implementazione software degli operatori matematici volti a sopprimere il gradiente isotropo della chioma.

Capitolo 2

Requisiti, Architettura e Design

Il presente capitolo si pone l'obiettivo di delineare l'infrastruttura ingegneristica e le scelte architetturali che costituiscono le fondamenta del software. Prima di addentrarsi nei dettagli implementativi legati all'interfaccia utente o alle librerie di calcolo, è indispensabile analizzare il contesto in cui l'applicazione si inserisce, definendo i requisiti di dominio, funzionali e prestazionali che hanno guidato l'intero ciclo di sviluppo.

2.1 Analisi dei Requisiti e Contesto di Dominio

Il progetto nasce da una reale e tangibile necessità della comunità astronomica: disporre di uno strumento software moderno, completo e altamente specializzato per l'elaborazione di immagini cometarie. Durante l'intera fase di ideazione e sviluppo, il lavoro è stato strettamente affiancato e supervisionato da ricercatori e astrofili affiliati all'Osservatorio Astrofisico di Asiago. Questa collaborazione diretta con gli utenti finali (*stakeholder*) ha permesso di far emergere le criticità tipiche del loro flusso di lavoro quotidiano, fornendo feedback continui e delineando un set di requisiti molto più rigoroso rispetto a quello dei software generalisti per l'astrofotografia.

L'analisi di dominio ha evidenziato come l'elaborazione delle comete richieda un approccio matematico e concettuale diverso rispetto a quello utilizzato per i classici corpi celesti. Infatti, data la natura dinamica delle comete e il vasto gradiente di segnale che separa il nucleo brillante dalle deboli emissioni gassose della chioma, si rende indispensabile l'impiego di algoritmi mirati e di una precisione di calcolo assoluta.

2.1.1 Gestione del Dato Scientifico e dei Metadati

Il primo requisito fondamentale emerso dai colloqui con i ricercatori riguarda la gestione del dato grezzo. In ambito scientifico, le immagini astronomiche vengono acquisite e salvate nel formato FITS (di cui si discuterà in modo approfondito nel Capitolo successivo). Questo formato non si limita a memorizzare una semplice griglia di pixel, ma costituisce un vero e proprio contenitore di dati scientifici complessi.

Un singolo file FITS di una ripresa cometaria può superare agevolmente le decine di megabyte di peso, in quanto i valori di intensità luminosa dei pixel sono spesso registrati in virgola mobile a doppia precisione (*Double*) per non perdere la minima variazione fotometrica. Inoltre, un'analisi statistica e morfologica non viene quasi mai condotta su un singolo scatto, ma su intere sequenze temporali che possono comprendere decine o centinaia di immagini alla volta. Il software deve pertanto garantire una gestione della memoria estremamente efficiente per non saturare le risorse del sistema durante il caricamento massivo e l'analisi di tali sequenze.

Parallelamente ai dati visivi, i file FITS ospitano un *Header* contenente i metadati dell'immagine: coordinate celesti, temperature del sensore, tempi di esposizione e, soprattutto, la storia delle elaborazioni subite. I ricercatori hanno sottolineato come un software scientificamente valido

non debba limitarsi a manipolare i pixel, ma debba obbligatoriamente tracciare ogni modifica. Si è reso quindi necessario sviluppare un sistema in grado di aggiornare dinamicamente l'header di ciascuna immagine elaborata, registrando accuratamente le operazioni compiute e i parametri utilizzati, al fine di garantire la totale riproducibilità e validità dell'analisi scientifica.

2.1.2 Requisiti Funzionali: Precisione e Specificità

A differenza dei programmi di post-produzione generalisti, il software doveva fornire una suite di strumenti pensata esclusivamente per la morfologia cometaria. Le operazioni tipiche richieste dal flusso di lavoro dei ricercatori comprendono: la ricerca automatizzata del nucleo, l'allineamento dell'intera sequenza sulla cometa o, alternativamente, inseguendo il campo stellare di fondo, la successiva fase di integrazione (*stacking*) e, infine, l'applicazione di filtri spaziali per l'evidenziazione delle strutture a basso contrasto (come getti di polveri).

Un aspetto critico emerso dall'analisi dello stato dell'arte riguarda l'estrema complessità algoritmica nell'identificazione del centroide del nucleo prima dell'applicazione dei filtri morfologici. Data la natura diffusa della chioma, ottenere una precisione sub-pixel costante è un'operazione delicata: anche una minima incertezza nel calcolo del centroide può distorcere i risultati dei filtri basati su trasformate radiali o rotazionali, introducendo artefatti che rischiano di essere scambiati per false strutture fisiche della cometa. L'algoritmo di rilevamento interno deve pertanto garantire una precisione assoluta per fungere da base affidabile a ogni successiva elaborazione.

Proprio in questa fase di analisi morfologica, il progetto ha assunto una forte connotazione sperimentale. Oltre all'implementazione dei filtri classici presenti in letteratura (come il gradiente rotazionale o il filtro di Larson-Sekanina), si è cercato di sviluppare e testare nuove soluzioni matematiche per l'enfatizzazione del segnale. L'obiettivo era individuare operatori d'immagine capaci di estrarre dettagli ancora più fini e meno evidenti, minimizzando al contempo l'introduzione di rumore computazionale.

Un'ulteriore esigenza funzionale, che ha portato allo sviluppo delle logiche esposte precedentemente, è la gestione del campo stellare. Nelle immagini cometarie, le stelle di fondo "sporcano" i calcoli fotometrici e, a seguito dell'allineamento sul nucleo in movimento, generano vistose strisce luminose (*star trails*) durante la fase di somma. È nato da qui il requisito di integrare un motore di mascheramento e rimozione stellare proprietario e altamente specifico, atto a isolare il segnale del target dalle sorgenti puntiformi estranee.

2.1.3 Requisiti Non Funzionali: Performance, Portabilità e Usabilità

Tutta l'accuratezza matematica richiesta dai ricercatori doveva però scontrarsi con la realtà dell'infrastruttura informatica. Il software doveva essere sviluppato tenendo a mente tre pilastri non funzionali:

1. **Performance:** Sviluppare algoritmi di altissima precisione matematica e morfologica senza appesantire il sistema o dilatare in modo inaccettabile i tempi di esecuzione, specialmente durante l'elaborazione batch di file di grandi dimensioni.

2. **Cross-Platform:** Data l'eterogeneità dei sistemi operativi in ambito scientifico e amatoriale, il software deve supportare flussi di lavoro distribuiti tra diverse piattaforme, garantendo una compilazione ed esecuzione nativa su tutti i principali ecosistemi (Windows, macOs, Linux).

3. **Usabilità Interattiva:** Storicamente, molti degli applicativi astronomici più consolidati presentano interfacce utente basate su paradigmi di interazione rigidi e sequenziali, che spesso riflettono flussi di lavoro lineari e poco flessibili. Tali soluzioni, pur essendo tecnicamente robuste, possono risultare poco intuitive e limitanti durante la fase esplorativa della ricerca morfologica, dove la possibilità di sperimentare variazioni parametriche in modo rapido è fondamentale. Si è deciso pertanto di dotare il software di una User Interface (UI), altamente reattiva e progettata secondo canoni moderni, che consenta un controllo dinamico e non

lineare del flusso di lavoro. L'obiettivo è offrire un ambiente operativo capace di fornire un riscontro visivo immediato, permettendo all'operatore di navigare tra le diverse fasi dell'analisi con una fluidità e una libertà di manovra superiori rispetto alle architetture software più datate del settore.

4. **Internazionalizzazione e Localizzazione:** Data la natura globale della ricerca scientifica e la frequente collaborazione tra team internazionali, il software deve essere progettato per agevolare la localizzazione dell'interfaccia in diverse lingue. Questo requisito è fondamentale per garantire l'accessibilità dello strumento a un numero di utenti maggiore, facilitando l'adozione di Kometra [3] in contesti accademici eterogenei.

2.2 Scelta dello Stack Tecnologico

La definizione dello stack tecnologico ha richiesto un'attenta ponderazione delle innumerevoli alternative offerte dal mercato software, costantemente guidata dai vincoli di prestazione, manutenibilità e usabilità descritti in fase di analisi dei requisiti. Costruire un'applicazione desktop per l'elaborazione di dati scientifici massivi impone infatti di scendere a compromessi tra la velocità pura dell'esecuzione nativa e la flessibilità di sviluppo dell'interfaccia grafica.

2.2.1 Il Linguaggio di Programmazione: Dal Prototipo a C#

Una prima fase esplorativa del progetto è stata condotta realizzando una demo sperimentale in linguaggio Python. Questo linguaggio rappresenta attualmente lo standard di fatto per la prototipazione e l'analisi dati in ambito astrofisico, grazie alla vastissima disponibilità di librerie scientifiche precompilate. Tuttavia, i test sul prototipo hanno evidenziato criticità insormontabili per un'applicazione desktop interattiva. Python, essendo un linguaggio interpretato e storicamente limitato dal *Global Interpreter Lock* per la gestione dei thread, mostrava una marcata lentezza di esecuzione nella manipolazione diretta di matrici di pixel ad alta risoluzione. Inoltre, l'ecosistema Python presenta un'estrema frammentazione e difficoltà ingegneristica nel confezionare un'interfaccia UI visiva che risulti realmente completa, moderna e responsiva.

Per superare i limiti prestazionali, l'analisi si è spostata sulla valutazione di linguaggi compilati a basso livello e ad altissime prestazioni, come C, C++ o Rust. Se da un lato questi strumenti garantiscono una velocità computazionale ineguagliabile e un controllo totale sulla memoria di sistema, dall'altro soffrono intrinsecamente di una forte mancanza di standardizzazione per quanto concerne lo sviluppo di interfacce grafiche multipiattaforma. Creare una UI complessa interamente in C++ avrebbe richiesto tempi di sviluppo incompatibili con le tempistiche del progetto, rendendo il codice inutilmente verboso. Allo stesso modo, l'idea di mantenere un comparto matematico scritto in Python o in C puro, per poi richiamarlo da un'interfaccia sviluppata in un altro linguaggio tramite script esterni, è stata categoricamente scartata. La continua traduzione, serializzazione e comunicazione dei dati tra i due ambienti avrebbe introdotto un *overhead* e una latenza inaccettabili per un software che deve operare fornendo un riscontro visivo in tempo reale all'operatore.

La scelta finale e definitiva per l'architettura core del software è ricaduta su C# e sull'ecosistema .NET. Questo linguaggio orientato agli oggetti di casa Microsoft offre infatti il compromesso ingegneristico perfetto per l'applicativo in esame. Grazie al suo compilatore e alle moderne ottimizzazioni del *runtime*, C# garantisce prestazioni di calcolo estremamente elevate, del tutto paragonabili a quelle dei linguaggi compilati a più basso livello, pur sollevando lo sviluppatore dalle insidie della gestione manuale della memoria grazie a un *Garbage Collector* altamente ottimizzato. Inoltre, il framework .NET vanta oggi uno degli ecosistemi più maturi, documentati e supportati a livello globale per la creazione di applicazioni desktop complesse e per la gestione del multithreading.

2.2.2 Il Framework Grafico: L'Adozione di Avalonia UI

Stabilito il linguaggio di programmazione, si è resa necessaria l'individuazione del framework più adatto per la costruzione dell'interfaccia grafica. L'analisi delle opzioni disponibili ha escluso in primo luogo le moderne soluzioni basate su tecnologie web e HyperText Markup Language (HTML) (come Electron o WebView). Sebbene queste architetture permettano di ottenere design esteticamente molto curati sfruttando le tecnologie del web design, esse introducono pesanti strati di astrazione: l'applicazione finisce per eseguire un intero motore browser in background. Questo causa un notevole calo prestazionale, una latenza di comunicazione interna e un consumo eccessivo di memoria Random Access Memory (RAM), difetti assolutamente fatali per un'applicazione desktop destinata a caricare decine di file FITS pesanti in memoria.

Si è poi valutata l'adozione di Windows Presentation Foundation (WPF), il framework storico e consolidato in ambito Microsoft per lo sviluppo desktop. WPF offre un sistema di *Data Binding* eccezionale e un controllo granulare sugli elementi visivi tramite il linguaggio di markup eXtensible Application Markup Language (XAML). Tuttavia, esso risulta ormai tecnologicamente datato e, soprattutto, è legato a doppio filo alle librerie grafiche DirectX, rendendolo compatibile esclusivamente con i sistemi operativi Windows. Inoltre, WPF è noto per avere una curva di apprendimento particolarmente ripida e complessa.

La soluzione ingegneristica ottimale è stata infine individuata in Avalonia UI [4]. Si tratta di un framework open source e gratuito che nasce con una vocazione puramente multipiattaforma. Avalonia eredita i concetti logici e strutturali più validi del mondo Microsoft, compreso l'uso di XAML, ma li modernizza profondamente. Offre una curva di apprendimento più dolce e si appoggia a motori di rendering grafici moderni e indipendenti dal sistema operativo, capaci di garantire interfacce fluide, accelerazioni hardware native e design al passo con i tempi. L'aspetto cruciale che ha determinato questa scelta è la possibilità di scrivere l'intero codice grafico e logico una sola volta, potendolo poi compilare ed eseguire in modo nativo e pixel-perfect su macchine Windows, macOS e distribuzioni Linux. Questo soddisfa a pieno i stringenti requisiti di portabilità richiesti dal dominio astronomico, dove hardware di acquisizione e hardware di elaborazione spesso differiscono.

2.2.3 Motore Computazionale: L'Integrazione di OpenCV

L'ultimo tassello dello stack tecnologico riguarda la gestione puramente matematica dei dati. Le immagini astronomiche sono, nella loro essenza, enormi matrici bidimensionali di numeri a virgola mobile. Per garantire la massima rapidità e precisione nei calcoli su queste strutture dati, si è scelto di non implementare la pura manipolazione dei pixel iterando con semplici cicli in codice C# nativo. L'accesso sequenziale a milioni di pixel in memoria gestita avrebbe comportato un inevitabile degrado delle prestazioni.

Lo sviluppo si è pertanto appoggiato a OpenCV [5], la libreria open source leader a livello mondiale per la visione artificiale e l'elaborazione di immagini. Sfruttando un apposito *wrapper* compatibile con l'ecosistema .NET, denominato OpenCvSharp [6], il software è in grado di dialogare fluidamente con la libreria sottostante. Questo stratagemma architetturale permette di delegare le complesse operazioni morfologiche, le trasformazioni spaziali, le binarizzazioni e i calcoli di algebra lineare direttamente alle funzioni native scritte in C e C++. Tali funzioni sono ottimizzate a livello di istruzioni del processore, sfruttando la vettorializzazione hardware e garantendo così velocità di esecuzione massimali. La stabilità pluridecennale, l'efficienza algoritmica e l'eccellente documentazione scientifica di OpenCV hanno permesso di implementare gli algoritmi di allineamento, mascheramento e ricostruzione stocastica garantendo un livello di affidabilità prestazionale pienamente conforme ai rigorosi standard imposti dalla ricerca astrofisica.

2.3 Architettura Software: Pattern Implementativi

2.3.1 Il Pattern Model-View-ViewModel (MVVM)

Per garantire un'elevata manutenibilità del codice e una netta separazione tra la logica applicativa e l'interfaccia grafica, l'intera architettura del software è stata modellata attorno al pattern architetturale Model-View-ViewModel (MVVM) [8]. Questo paradigma, ideato da Microsoft e oggi standard di fatto per le moderne applicazioni desktop, permette di disaccoppiare completamente il livello di presentazione dal dominio dei dati. I vantaggi di questo approccio sono molteplici: agevola la scrittura di unit test, favorisce la riusabilità del codice e permette a sviluppatori o designer di lavorare sull'interfaccia visiva senza intaccare le funzionalità matematiche o logiche sottostanti.

Dal punto di vista puramente teorico, il pattern si fonda su tre componenti logici governati da una rigida regola di dipendenza unidirezionale: la View conosce il ViewModel, il ViewModel conosce il Model, ma il Model è del tutto ignaro dell'esistenza degli strati superiori. La comunicazione "dal basso verso l'alto" non avviene mai tramite chiamate dirette, bensì sfruttando il pattern comportamentale *Observer* [9]. Quando un dato nel ViewModel cambia, esso emette un evento di notifica; la View, che è in ascolto, intercetta l'evento e si aggiorna di conseguenza. Parimenti, le azioni dell'utente sulla View non invocano metodi diretti, ma attivano dei *Command* (comandi astratti) esposti dal ViewModel.

Nel contesto specifico di questa applicazione, il livello dei *Model* è stato implementato seguendo il paradigma dei Plain Old CLR Object (POCO). I modelli sono classi estremamente leggere, il cui unico scopo è detenere lo stato persistente e i dati strutturali dell'applicazione. Questa scelta progettuale garantisce che tali oggetti siano agnostici rispetto al framework di presentazione, rendendoli ideali per le operazioni di persistenza e serializzazione. In questi oggetti è del tutto assente qualsiasi tipo di logica elaborativa o riferimento a librerie grafiche, rispettando rigorosamente il principio di isolamento del dominio.

A fare da ponte logico tra i dati grezzi e l'interfaccia intervengono i *ViewModel*. Ogni finestra o componente visivo del programma possiede un proprio ViewModel dedicato, che funge da astrazione dello stato della vista. Il suo compito è formattare i dati del Model in un formato facilmente digeribile dall'interfaccia ed esporre i *Command* per le interazioni. Per l'implementazione di questo strato, Kometra [3] sfrutta la libreria `CommunityToolkit.Mvvm`, la quale, attraverso l'ereditarietà dalla classe `ObservableObject` e l'uso di attributi a compilazione sorgente (*Source Generators*) come `[ObservableProperty]` e `[RelayCommand]`, riduce drasticamente il codice ripetitivo. Questo sistema genera in automatico l'implementazione dell'interfaccia standard `INotifyPropertyChanged`, scatenando gli eventi di notifica necessari ad aggiornare l'interfaccia utente ogni qualvolta un dato cambi valore.

L'ultimo livello del pattern è costituito dalle *View*, supportate nativamente in Avalonia UI tramite il suo potente motore di *Data Binding*. Le View sono composte da un file di markup dichiarativo in linguaggio XAML e dal relativo *code-behind* in C#. Nel file XAML viene definita la gerarchia visiva dei controlli, sfruttando il *Data Binding* per collegare dinamicamente e in modo dichiarativo gli elementi grafici alle proprietà esposte dal ViewModel. Il *code-behind* delle viste, che nei pattern architetturali più puristi andrebbe mantenuto vuoto, in un'applicazione fortemente interattiva come questa ricopre invece un ruolo essenziale e ben delimitato: esso è stato infatti riservato in modo esclusivo alla gestione diretta degli eventi hardware del puntatore, operazioni spaziali difficilmente traducibili in dichiarazioni XAML pure.

Infine, per soddisfare i requisiti di localizzazione definiti precedentemente, la View non contiene stringhe testuali statiche; ogni etichetta e messaggio dell'interfaccia è gestito tramite file di risorse (.resx). Questo approccio permette di implementare nativamente il supporto bilingue (italiano e inglese) e garantisce la futura estensibilità del software verso ulteriori lingue attraverso la semplice aggiunta di nuovi dizionari, senza richiedere alcuna modifica alla logica applicativa.

2.3.2 Inversion of Control e Dependency Injection

L'astrazione fornita dal pattern MVVM risulterebbe incompleta e fragile se i ViewModel e i Coordinatori dovessero istanziare manualmente i propri motori di calcolo tramite costruttori espliciti. Questo approccio tradizionale genererebbe un forte accoppiamento (*tight coupling*) tra le classi, rendendo il codice rigido, difficile da testare e complesso da aggiornare. Per ovviare a questo problema concettuale, l'intera architettura del software Kometra [3] si fonda sul principio dell'IoC, implementato tecnicamente attraverso la Dependency Injection (DI) [10].

Per la gestione delle dipendenze è stato adottato il container standard dell'ecosistema .NET (`Microsoft.Extensions.DependencyInjection`). Il punto di ingresso dell'applicazione, definito nella classe principale `App.cs`, agisce come *Composition Root*: è il luogo centrale e unico in cui il software, al momento dell'avvio, configura un registro globale conoscendo e istanziando tutte le astrazioni e le relative implementazioni concrete.

L'analisi del metodo di configurazione dei servizi, illustrato nella Figura 2.1, evidenzia una rigorosa classificazione strutturale divisa per domini di competenza. Il registro inietta innanzitutto i servizi di infrastruttura e Input/Output (I/O), come la gestione delle finestre di dialogo (`DialogService`) e dell'interfaccia (`WindowService`). A questi seguono i moduli per la gestione dei dati grezzi e dei metadati FITS, i motori scientifici e di rendering (come `RadiometryEngine` e `SegmentationEngine`), i coordinatori operativi e, infine, i *ViewModel* principali.

```
private static IServiceProvider ConfigureServices()
{
    var services = new ServiceCollection();

    // Infrastruttura e I/O
    services.AddSingleton<IDialogService, DialogService>();
    services.AddSingleton<IWindowService, WindowService>();

    // Dati e Metadati FITS
    services.AddSingleton<IFitsDataManager, FitsDataManager>();
    services.AddSingleton<IFitsOpenCvConverter, FitsOpenCvConverter>();

    // Motori Scientifici e di Elaborazione
    services.AddSingleton<IRadiometryEngine, RadiometryEngine>();
    services.AddSingleton<ISegmentationEngine, SegmentationEngine>();
    services.AddSingleton<IInpaintingEngine, InpaintingEngine>();

    // Coordinatori
    services.AddSingleton<IMaskingCoordinator, MaskingCoordinator>();
    services.AddSingleton<IStackingCoordinator, StackingCoordinator>();

    // ViewModels Principali
    services.AddSingleton<MainWindowViewModel>();
    services.AddSingleton<BoardViewModel>();

    return services.BuildServiceProvider();
}
```

Figura 2.1. Estratto della classe `App.cs` che funge da Composition Root per la registrazione dei servizi nel container IoC.

Un aspetto progettuale di rilievo riguarda la scelta del ciclo di vita (*lifecycle*) dei servizi registrati. Come visibile dal codice, la quasi totalità dei motori matematici e dei coordinatori è stata registrata come `Singleton`. Questa scelta permette di ottimizzare l'occupazione della RAM per i moduli *stateless*, mentre le preferenze dell'utente tra diverse sessioni di lavoro vengono preservate attraverso la `ToolParametersCache`. Tale architettura consente di mantenere le finestre di

dialogo rigorosamente `Transient`: l'interfaccia può così essere distrutta e ricreata a ogni invocazione, garantendo una gestione pulita delle risorse grafiche senza perdere la continuità dei dati, che rimangono persistenti all'interno della cache centralizzata.

Infine, l'architettura dimostra una notevole flessibilità nella gestione dei ViewModel legati agli strumenti operativi specifici (i *Tool ViewModels*, come quelli dedicati all'esportazione o all'allineamento). Tali componenti non vengono registrati nel container globale all'avvio. La motivazione ingegneristica risiede nel fatto che questi ViewModel necessitano di dati dinamici risolvibili esclusivamente a tempo di esecuzione (*runtime*), come i percorsi dei file selezionati istantaneamente dall'utente sulla Board. Per questi casi specifici, il software adotta un approccio ibrido: delega la loro istanziazione al `WindowService`, il quale provvede a creare le istanze manualmente iniettandovi i dati di contesto necessari, mantenendo al contempo immutata la purezza del container principale.

2.3.3 Applicazione Pratica: L'Architettura dei Nodi e l'Asincronia

Per comprendere appieno l'applicazione pratica di questi pattern, è possibile analizzare il caso dei nodi, gli elementi atomici che compongono la Board di Kometra (i quali verranno approfonditi nella sezione successiva). All'interno del layer di modellazione, classi come `BaseNodeModel` e le sue derivazioni (`SingleImageNodeModel` e `MultipleImagesNodeModel`) contengono esclusivamente proprietà primarie: un identificativo univoco, le coordinate spaziali sulla Board, un titolo e i percorsi testuali dei file FITS associati. In questi modelli è del tutto assente qualsiasi tipo di logica elaborativa o riferimento a librerie grafiche.

Un aspetto cruciale dell'architettura riguarda l'aderenza al Principio di Singola Responsabilità (Single Responsibility Principle) [11]. Per evitare di appesantire i ViewModel trasformandoli in classi onniscienti e monolitiche, l'intera logica di dominio e di elaborazione è stata estratta e confinata in servizi separati. Il ViewModel di un nodo multiplo, ad esempio, non contiene alcun codice capace di aprire un file FITS o di manipolarne i pixel. Al contrario, esso riceve le proprie dipendenze tramite iniezione (DI), interfacciandosi con servizi specializzati come `IFitsDataManager` e `IFitsRendererFactory`. Il compito del ViewModel si riduce quindi a una pura orchestrazione asincrona: quando l'utente cambia immagine nella sequenza, il ViewModel istanzia un *CancellationToken* per gestire eventuali interruzioni, invoca il servizio di caricamento in un Task asincrono separato (`Task.Run`) per non bloccare il thread dell'interfaccia, e infine espone alla grafica il renderer aggiornato.

Il ciclo si chiude con lo strato della View, dove la definizione dell'interfaccia convive con la propria logica di presentazione (*code-behind*). È proprio in quest'ultima che vengono intercettati gli eventi hardware di basso livello: il *code-behind* gestisce il trascinamento fisico (*drag*) del nodo ricalcolandone le matrici di trasformazione sulla Board o cattura lo scorrimento della rotella del mouse per alterare dinamicamente lo zoom. Queste interazioni puramente grafiche vengono poi sincronizzate con il ViewModel tramite *Data Binding* o chiamate a comandi, garantendo un'esperienza fluida. Questo approccio permette alla View di reagire istantaneamente agli input dell'utente.

2.4 Interfaccia Utente: Spazio di Lavoro e Flusso a Nodi

Oltre all'ottimizzazione del motore computazionale, una delle sfide ingegneristiche più complesse ha riguardato la progettazione dell'interfaccia utente. L'obiettivo era superare i limiti dei tradizionali software di elaborazione fotografica, i quali si affidano tipicamente a un approccio lineare basato su uno storico delle azioni sequenziale o su una rigida struttura a livelli. Per l'analisi scientifica delle comete, è essenziale mantenere una visibilità totale su tutte le operazioni eseguite a partire dai dati grezzi, conservando sia una cronologia chiara delle trasformazioni, sia la possibilità di creare ramificazioni per esplorare parametri operativi alternativi e confrontarne visivamente i risultati.

Per rispondere a questa esigenza, si è scelto di implementare un'interfaccia basata su un paradigma a nodi immersi in uno spazio di lavoro denominato Board. La Board costituisce

un'area di lavoro bidimensionale virtualmente infinita, priva di confini fisici, all'interno della quale l'utente può muoversi in totale libertà. La navigazione all'interno di questo spazio avviene tramite operazioni fluide di traslazione e ingrandimento, permettendo all'operatore di organizzare visivamente le proprie immagini senza doversi preoccupare di sforare alcun limite spaziale imposto dallo schermo.

All'interno di questo spazio infinito prendono posto i nodi. È fondamentale chiarire una distinzione architetturale rispetto ai classici applicativi nodali: mentre in molti software di computer grafica un nodo rappresenta un blocco matematico che esegue un'operazione specifica trasformando un input in un output, in Kometra [3] il paradigma è diverso. In questo software, il nodo funge essenzialmente da contenitore di dati e di stati visivi. Esso può essere posizionato e spostato a piacimento sulla Board e su di esso vengono successivamente invocate le operazioni di elaborazione tramite i menu contestuali o la barra degli strumenti superiore.

I nodi implementati si dividono in due categorie strutturali: nodi singoli e nodi multipli. Un nodo singolo ospita una singola immagine, mentre un nodo multiplo è progettato per contenere un'intera sequenza temporale di fotogrammi riferiti alla medesima cometa. Questa seconda tipologia risulta cruciale per abbattere i tempi operativi, in quanto permette al sistema di applicare una trasformazione geometrica o morfologica in modalità batch a tutte le immagini contenute al suo interno, evitando all'utente la tediosa ripetizione dei medesimi passaggi per ogni singolo scatto. L'interfaccia del nodo multiplo integra inoltre controlli dedicati per scorrere rapidamente l'intera sequenza caricata.

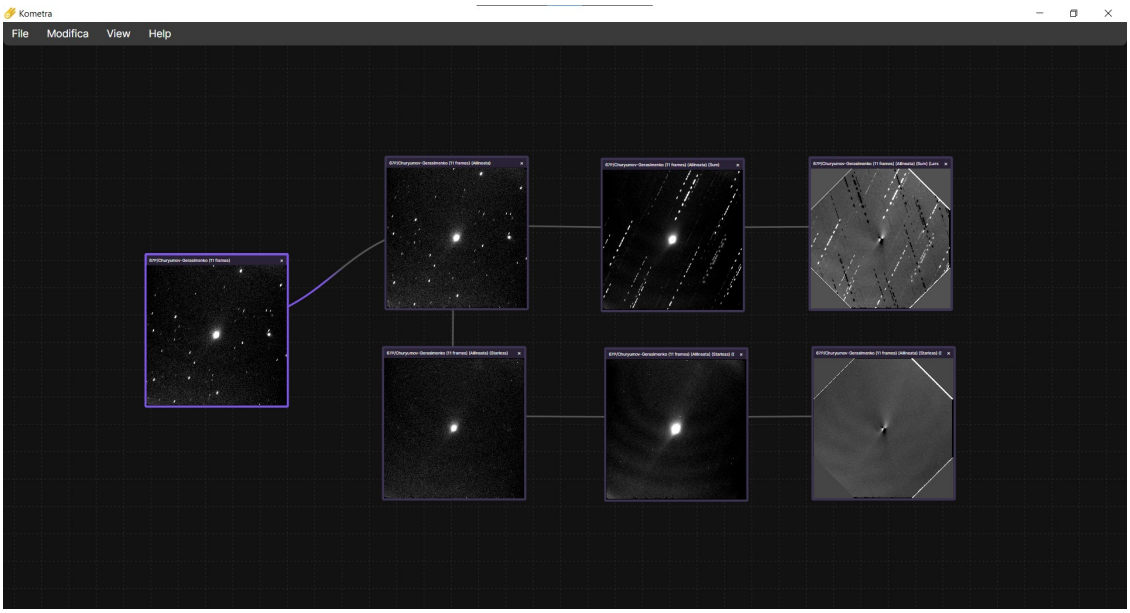

Figura 2.2. Panoramica della Board di Kometra [3]: è visibile architettura a nodi interconnessi per un workflow non distruttivo basato sulla generazione di nodi figli dai dati grezzi.

Ogni contenitore nodale offre un elevato grado di interattività. L'utente può rinominare liberamente il nodo per documentare le proprie fasi di analisi, nonché ingrandire o traslare l'immagine contenuta lavorando direttamente all'interno della cornice del nodo stesso. L'interazione più critica riguarda tuttavia la manipolazione delle soglie di visualizzazione, controllabile dinamicamente tramite la rotella del mouse. Questa funzionalità è resa strettamente necessaria dalla natura stessa del formato FITS. I sensori astronomici acquisiscono dati lineari, distribuendo il segnale luminoso su ampiezze a sedici, trentadue o sessantaquattro bit. I monitor standard e l'occhio umano, tuttavia, non sono in grado di percepire questa vasta gamma dinamica lineare; di conseguenza, un'immagine astronomica grezza appare tipicamente come un rettangolo quasi del tutto nero, in cui solo i nuclei stellari saturi risultano debolmente visibili. Per poter ispezionare visivamente le tenui strutture cometarie senza alterare i dati scientifici sottostanti, è necessario applicare una funzione di trasferimento a schermo. Regolando le soglie di visualizzazione (il punto di nero e il punto di bianco), l'operatore comprime visivamente l'istogramma, rivelando il segnale nascosto

nel rumore di fondo. Il nodo permette di compiere questa operazione in tempo reale, garantendo un'ispezione ottimale del dato.

L'intero flusso di lavoro all'interno della Board è stato infine vincolato a una rigorosa filosofia non distruttiva. I dati astronomici originali non devono mai essere sovrascritti o alterati accidentalmente, per scongiurare la perdita irrimediabile del rigore scientifico dell'acquisizione. Pertanto, i nodi di base agiscono in sola lettura sui file sorgenti. Ogniqualvolta l'utente seleziona un nodo e avvia una trasformazione tramite il menu che altera fisicamente i pixel o la geometria dell'immagine, il software elabora l'operazione e genera automaticamente un nuovo nodo figlio, collegandolo visivamente al nodo padre. Questo nuovo contenitore ospita il risultato dell'elaborazione, mantenendo i dati temporaneamente nella memoria volatile del sistema. Alla chiusura dell'applicativo, tali dati temporanei vengono scartati, a meno che l'operatore non decida esplicitamente di esportare e salvare i risultati consolidati in un nuovo percorso di archiviazione.

Capitolo 3

Infrastruttura Dati e Standard FITS

Entriamo ora più nel dettaglio di come è necessario gestire i dati per quanto riguarda l'imaging astronomico. In particolare, nello studio delle comete il concetto di "immagine" differisce sostanzialmente da quello utilizzato nella fotografia tradizionale: mentre i formati standard di consumo (come Joint Photographic Experts Group (JPEG) o Portable Network Graphics (PNG)) sono ottimizzati per la percezione visiva umana, un'immagine astronomica costituisce a tutti gli effetti una misura fisica bidimensionale e l'applicazione di strategie di compressione *lossy* equivarrebbe alla perdita di dati importantissimi.

Ogni pixel del sensore Charge-Coupled Device (CCD) o Complementary Metal-Oxide Semiconductor (CMOS) non rappresenta un semplice colore, bensì un conteggio di fotoni (o meglio, di elettroni convertiti in ADUs) accumulati durante il tempo di esposizione. Affinché l'analisi scientifica sia valida, è imperativo mantenere la rigorosa linearità del segnale e preservare l'intera gamma dinamica del sensore, che quasi sempre eccede i classici 8 bit per canale, arrivando a 16 o 32 bit (interi o con virgola mobile).

Lo standard universale adottato dalla comunità scientifica per veicolare queste informazioni è il formato FITS. A differenza di un semplice contenitore di pixel, il FITS incapsula in un unico file sia la matrice dei dati grezzi (Raw Data) sia un ampio corredo di metadati (Header), essenziali per descrivere le condizioni di acquisizione (coordinate celesti, temperatura del sensore, tempo di posa, strumentazione ottica).

Le sfide principali che sono sorte durante la progettazione di questa architettura superano la semplice visualizzazione del dato e integrità del dato scientifico. In primo luogo, il software deve anche assicurare la preservazione dei metadati, mantenendo le informazioni dell'Header (coordinate, tempo, ottica) inscindibilmente legate alla matrice dei pixel durante tutto il ciclo di vita dell'applicazione. Parallelamente, bisogna gestire con attenzione l'efficienza delle risorse e la gestione della memoria; infatti, l'analisi cometaria richiede tipicamente l'elaborazione di serie temporali o lo *stacking* di numerose immagini ad alta risoluzione.

Infine, il sistema di I/O è stato ottimizzato per gestire in modo trasparente anche i file compressi, implementando per il formato FITS strategie dedicate che si discostano dalle tecniche di decompressione tradizionale.

Nelle sezioni successive verranno analizzati i dettagli implementativi di questo flusso, partendo dall'architettura tecnica dello standard FITS e dalla sua modellazione tramite un'infrastruttura di servizi software dedicata. Verranno approfondite le strategie di gestione della memoria e le ottimizzazioni a basso livello necessarie per la manipolazione di grandi dataset binari, fino alla descrizione del sistema di compressione a tasselli (*Tiled Compression*) e allo sviluppo del motore di codifica *bit-exact* implementato per garantire efficienza e piena interoperabilità scientifica.

3.1 Il Formato FITS: Standard e Parsing

Nel panorama delle scienze fisiche e astronomiche, il formato FITS rappresenta lo standard globale per il trasporto, l'analisi e l'archiviazione dei dati. Infatti, il FITS nasce con una missione specifica:

la precisione scientifica assoluta e la conservazione dei metadati multidimensionali. La sua filosofia portante è poi racchiusa nel principio cardinale "Once FITS, Always FITS" [12].

Questo fondamento assicura che un dato acquisito in passato possa essere decodificato dai software odierni senza alcuna perdita di informazione. Tale rigorosa retro-compatibilità, promossa dall'Unione Astronomica Internazionale (IAU), non rappresenta un mero dettaglio tecnico, bensì un requisito imprescindibile per la ricerca astrofisica, che spesso richiede analisi comparative su scale temporali decennali.

Nello specifico dominio dell'astronomia cometaria, l'adozione di questo formato risulta determinante. A differenza delle sorgenti stellari di fondo, le comete sono oggetti intrinsecamente dinamici e transitori: la loro posizione orbitale e la morfologia della chioma evolvono rapidamente. Di conseguenza, il file non può limitarsi a memorizzare la pura matrice dei pixel per l'analisi fotometrica, ma deve obbligatoriamente incapsulare il preciso contesto astrometrico e temporale al momento dell'acquisizione.

3.1.1 Genesi Storica e Contesto Tecnologico

Procediamo quindi con una rapida contestualizzazione dell'epoca tecnologica in cui questo formato è stato concepito; infatti, le decisioni prese alla fine degli anni '70 per superare i limiti dell'hardware dell'epoca hanno gettato le basi per un'architettura che garantisce ancora oggi l'accessibilità dei dati astrofisici.

Il formato FITS è stato ideato nel 1981 da Don Wells ed Eric Greisen (National Radio Astronomy Observatory (NRAO)), in collaborazione con Ron Harten della Netherlands Foundation for Radio Astronomy [13]. L'obiettivo primario dei fondatori non era inizialmente l'imaging ottico, bensì la creazione di un protocollo per lo scambio di dati interferometrici tra diversi osservatori radio, facilitando in particolare il trasferimento di informazioni tra il Very Large Array (VLA) e il Westerbork Synthesis Radio Telescope. In quel periodo, la sfida principale per la comunità scientifica era l'assoluta mancanza di interoperabilità tra le architetture dei mainframe. I computer di allora (IBM, CDC, DEC VAX) utilizzavano lunghezze di parola e rappresentazioni numeriche differenti (a 16, 32 o 60 bit), rendendo il trasferimento di dati un'operazione complessa e soggetta a errori di interpretazione.

La decisione di strutturare il file FITS in record logici di 2880 byte fu una risposta diretta alle specifiche dei nastri magnetici a 9 tracce, il supporto di memorizzazione universale del tempo. Infatti, un blocco di tali dimensioni era sufficientemente piccolo da essere gestito dalle limitate memorie RAM dei minicomputer, ma abbastanza grande da permettere un trasferimento efficiente dai lettori di nastro. Ad oggi, sebbene i moderni sistemi di archiviazione digitale abbiano superato questi limiti fisici, il FITS preserva questa struttura modulare.

Il riconoscimento del FITS come "lingua franca" dell'astronomia avvenne ufficialmente nel 1982, quando l'Unione Astronomica Internazionale (IAU) lo adottò come standard unico per l'interscambio dei dati. Dal 1988 la supervisione del formato è nelle mani dell'IAU FITS Working Group (IAU-FWG): un comitato tecnico permanente che garantisce l'evoluzione dello standard proteggendone al contempo l'integrità.

L'evoluzione di questo specifico formato è stata guidata da un approccio estremamente conservativo, mirato a preservare l'integrità storica degli archivi astronomici. Dalla sua prima codifica formale, lo standard ha subito pochissime revisioni, passando dall'adeguamento per il formato data Year 2000 (Y2K) all'integrazione strutturale del WCS. L'iter evolutivo è culminato nell'attuale Versione 4.0 (2016) [1], che ha introdotto il supporto nativo per le coordinate temporali, colmando una lacuna fondamentale per la caratterizzazione di fenomeni transitori e oggetti dinamici come le comete.

Parallelamente al "Core Standard" definito rigorosamente dall'IAU, la flessibilità del formato è garantita da un ecosistema di "Convenzioni Registrate" [14]. Si tratta di estensioni tecniche (nuove keyword o strutture tabulari) sviluppate per rispondere a esigenze specifiche della strumentazione moderna che il documento base, concepito decenni fa, non poteva prevedere. Sebbene queste convenzioni siano ufficialmente catalogate, il loro utilizzo introduce un limite di interoperabilità:

non facendo parte dello standard obbligatorio, il loro supporto non è garantito universalmente da tutti i parser o software di analisi. Questo crea una potenziale frammentazione, in cui un file FITS rimane tecnicamente leggibile nella sua struttura di base, ma le informazioni avanzate codificate tramite convenzioni specifiche potrebbero non essere interpretate correttamente da software che aderiscono strettamente solo allo standard Core.

3.1.2 Architettura Tecnica: Header Data Units (HDUs) e Record Logici

Un file FITS non deve essere considerato come un flusso continuo di dati, bensì come una sequenza ordinata di unità denominate Header Data Units (HDUs). Ogni file inizia obbligatoriamente con un Primary HDU, il quale può essere seguito da un numero arbitrario di estensioni (Extension HDUs), contenenti tabelle o array di dati supplementari.

L'unità atomica di input/output in un file FITS è il record logico di 2880 byte; quindi, sia le intestazioni (Header) che le unità di dati (Data Unit) devono occupare un numero intero di blocchi di questa dimensione. Le caratteristiche delle due tipologie di blocchi sono le seguenti:

- **Header Blocks:** contengono i metadati in formato American Standard Code for Information Interchange (ASCII). Qualora le informazioni dell'header occupino solo una porzione del blocco, lo spazio rimanente deve essere riempito con spazi ASCII (codice esadecimale `0x20`) [15].
- **Data Blocks:** contengono i dati binari (pixel o tabelle). Se l'array di dati non riempie completamente l'ultimo blocco da 2880 byte, lo spazio residuo deve essere colmato con zeri nulli (`0x00`) o spazi, a seconda della tipologia di estensione [15].

La sezione dell'Header è composta da una sequenza di record a lunghezza fissa di 80 caratteri, definiti storicamente "card images" in omaggio alle schede perforate a 80 colonne [1]. Tale architettura a dimensione costante, che garantisce un accesso diretto e ottimizzato ai metadati mediante il calcolo dei relativi *offset* di memoria, obbedisce a una sintassi ben precisa:

- **Keyword (Byte 1-8):** Stringa alfanumerica in lettere maiuscole, allineata a sinistra qualora sia inferiore agli 8 caratteri.
- **Indicatore di Valore (Byte 9-10):** Deve contenere la sequenza `= ` (uguale seguito da uno spazio).
- **Value (Byte 11-80):** Campo dedicato al parametro vero e proprio. Nello specifico, le stringhe testuali devono essere racchiuse tra apici singoli (`'...'`) mentre i valori booleani sono espressi dai caratteri `T` (True) o `F` (False).
- **Commento:** Campo opzionale collocato nello spazio rimanente e separato dal valore tramite il carattere *slash* (`/`).

Per un'immagine cometaria, le keyword obbligatorie nel Primary HDU definiscono la geometria fondamentale del dato: `SIMPLE` (conformità), `BITPIX` (profondità di bit), `NAXIS` (numero di dimensioni) e `NAXISn` (estensione di ogni dimensione). Infine, c'è `END` che è l'unica keyword priva di valore che marca la fine logica dell'header [16].

Dopo aver fornito queste informazioni di base possiamo passare alla descrizione del codice sviluppato per gestire correttamente tutte queste caratteristiche. Sono quindi state definite apposite classi che modellano lo standard FITS in memoria; queste strutture sono progettate per essere agnostiche rispetto alla logica di business, fungendo da contenitori puri per le informazioni.

- **FitsHdu (Header Data Unit):** Rappresenta la singola unità logica dello standard. Essa associa indissolubilmente l'oggetto `FitsHeader` (i metadati) con l'array `PixelData` (la matrice dei dati grezzi).

- **FitsHeader:** Modella la struttura a dizionario dei metadati appena descritta. Questa classe gestisce al suo interno una collezione di oggetti `FitsCard`, preservandone rigorosamente l'ordine di inserimento e i commenti originali, elementi essenziali per mantenere la leggibilità umana richiesta dallo standard.
- **FitsCard:** Rappresenta la singola entry (o record) dei metadati. Modella la struttura della riga d'informazione contenente la keyword, il valore associato e l'eventuale commento, garantendo la conformità ai requisiti sintattici del formato.

Le istanze di queste classi vengono create e popolate attraverso tre servizi principali che agiscono a basso livello e permettono di tradurre i byte in dati utilizzabili. Alla base dello stack operano quindi `FitsReader` e `FitsWriter`:

- **FitsReader:** scansiona i blocchi sequenziali da 2880 byte, gestisce l'allineamento della memoria in lettura e risolve le criticità legate all'*Endianness* (conversione da Big-Endian a Little-Endian) e ai tipi numerici primitivi.
- **FitsWriter:** si occupa della serializzazione, garantendo il corretto padding dei blocchi con spazi o zeri e la formattazione rigorosa delle stringhe ASCII a 80 caratteri in scrittura.

Tuttavia, il resto dell'applicazione non interagisce mai direttamente con il Reader o il Writer, ma si rivolge a una facade per caricare o salvare i dati: `FitsIoService`. Il suo compito è coordinare l'apertura degli stream sicuri e gestire in modo trasparente la complessità dei file compressi (che descriveremo meglio più avanti), presentando all'esterno un'interfaccia unificata per l'accesso al disco.

3.1.3 Rappresentazione Binaria, Tipi di Dato e Gestione Memoria

Un aspetto critico che distingue nettamente l'imaging astronomico dalla fotografia tradizionale risiede nell'impatto dimensionale e nella complessità della rappresentazione binaria. A differenza dei formati commerciali (JPEG, PNG), tipicamente vincolati a una profondità fissa di 8 bit, lo standard FITS è progettato per preservare l'intera gamma dinamica dei sensori scientifici, supportando formati numerici altamente eterogenei.

Questa versatilità architetturale comporta due conseguenze dirette. In primo luogo, impone un'allocazione dinamica della memoria: il *parser* non può istanziare la matrice dei dati a priori, ma deve attendere la decodifica della keyword `BITPIX` per identificare il tipo primitivo e dimensionare correttamente le risorse, scongiurando *overflow* o allocazioni sovradimensionate. In secondo luogo, genera un drastico incremento del volume dei dati, poiché il passaggio a 16 o 32 bit moltiplica linearmente l'impronta in memoria. Considerando le risoluzioni delle moderne camere CMOS, un singolo scatto non compresso può superare agevolmente le decine di megabyte; di conseguenza, durante l'elaborazione di estese serie temporali, la saturazione della RAM diventa il principale collo di bottiglia computazionale.

I tipi di dato primitivi supportati nativamente dallo standard, determinati dal valore assunto dalla keyword `BITPIX`, sono riassunti nella Tabella 3.1.

BITPIX	Tipo di Dato FITS	C/C++	Nota Tecnica
8	Unsigned Byte	`unsigned char`	Nessun byte swap richiesto.
16	Signed 16-bit Integer	`short`	Richiede byte swapping.
32	Signed 32-bit Integer	`int`	Richiede byte swapping.
64	Signed 64-bit Integer	`long long`	Introdotto nelle versioni recenti.
-32	IEEE-754 Float	`float`	Standard IEEE floating point.
-64	IEEE-754 Double	`double`	Standard IEEE floating point.

Tabella 3.1. Tipi di dato supportati dallo standard FITS in base al valore di BITPIX.

Analizzando la tabella emerge tuttavia una limitazione storica significativa: lo standard FITS non supporta nativamente gli interi senza segno a 16 o 32 bit. Questa carenza rappresenta un problema non trascurabile, poiché proprio questi formati (Unsigned 16-bit Integer) costituiscono l'output nativo della quasi totalità dei convertitori Analogico-Digitale (Analog-to-Digital Converters (ADCs)) presenti nelle camere CCD e CMOS per astronomia.

Per ovviare a questa limitazione senza violare lo standard, sono state introdotte le keyword `BSCALE` (fattore di scala) e `BZERO` (offset) che permettono di effettuare una trasformazione lineare dei dati in fase di lettura e scrittura:

$$Valore_{fisico} = (Valore_{FITS} \times BSCALE) + BZERO$$

Concretamente, per memorizzare i dati di un sensore a 16-bit unsigned (che variano nel range $[0, 65535]$), i valori vengono salvati nel file come signed 16-bit (range $[-32768, +32767]$), impostando nell'header:

- `BZERO = 32768`
- `BSCALE = 1.0`

Questa operazione di "shift" matematico permette al software di lettura di ricostruire i conteggi fotonici originali (ADUs) in modo trasparente per l'utente, garantendo la piena compatibilità con le specifiche IAU [1].

Va però precisato che l'utilizzo di questa coppia di keyword non è limitato alla sola gestione degli interi senza segno: questa trasformazione viene frequentemente impiegata anche per convertire i conteggi grezzi in unità fisiche (calibrazione) o per ottimizzare lo spazio di archiviazione, salvando valori in virgola mobile come interi scalati.

Infine, un'ulteriore criticità architetturale è rappresentata dalla gestione dell'*Endianness*. Infatti, lo standard FITS impone il Big-Endian (Network Byte Order), mentre le moderne Central Processing Units (CPUs) operano in Little-Endian. Ciò obbliga il software di lettura a eseguire un byte-swapping per ogni singolo valore numerico multibyte, pena la completa corruzione del dato.

A livello implementativo, dato il peso delle matrici descritte sopra, l'architettura evita il caricamento prematuro dei dati. In particolare, sono state definite le seguenti due classi:

- **FitsFileReference**: è un oggetto che rappresenta un singolo file e che, invece di mantenere l'intero dataset costantemente in RAM (operazione onerosa per grandi immagini astronomiche), conserva il percorso del file e un riferimento all'*header volatile*. Quest'ultimo consiste nella copia dei metadati su cui il software opera le modifiche, permettendo di tracciare i cambiamenti in tempo reale senza dover caricare o riscrivere prematuramente la pesante matrice dei pixel.
- **FitsDataManager**: rappresenta il servizio di alto livello a cui i *ViewModel* o altri componenti si rivolgono per l'accesso ai dati delle immagini. Questo componente non si limita a smistare le richieste al `FitsIoService`, ma implementa una logica di *caching* intelligente: se un'immagine è richiesta frequentemente, il manager restituisce l'istanza già presente in memoria, evitando costosi accessi al disco e ottimizzando sia i tempi di risposta che l'occupazione della RAM.

Invece, la gestione dell'eterogeneità dei dati è centralizzata nella classe `FitsReader` già accennata precedentemente. Il metodo `ReadMatrix` agisce come un *dispatcher*: ispeziona la keyword `BITPIX` e instrada il flusso di esecuzione verso il lettore specifico. Per garantire le massime prestazioni durante la lettura di matrici che possono contenere milioni di pixel, il codice evita calcoli matematici complessi all'interno dei cicli di lettura. In particolare, la conversione dell'*Endianness* è delegata alle primitive di sistema (`BinaryPrimitives`), che sfruttano le istruzioni CPU ottimizzate per invertire l'ordine dei byte in un singolo ciclo di clock.

```
public Array ReadMatrix(Stream stream, FitsHeader header) {
    int bitpix = GetInternalInt(header, "BITPIX", 0);
    int width = GetInternalInt(header, "NAXIS1", 0);
    int height = GetInternalInt(header, "NAXIS2", 0);
    double bzero = GetInternalDouble(header, "BZERO", 0.0);
    double bscale = GetInternalDouble(header, "BSCALE", 1.0);

    // Gestione immagini vuote (es. Primary Header senza dati)
    if (width <= 0 || height <= 0) {
        SkipDataBlock(stream, header);
        return Array.CreateInstance(typeof(byte), 0, 0);
    }

    Array result;

    // Selezione del tipo con gestione Unsigned (BZERO shift)
    if (bitpix == 8) {
        result = ReadBytePixels(stream, width, height);
    }
    else if (bitpix == 16) {
        if (IsUnsigned16(bzero, bscale))
            result = ReadUShortPixels(stream, width, height);
        else
            result = ReadInt16Pixels(stream, width, height);
    }
    else if (bitpix == 32) {
        if (IsUnsigned32(bzero, bscale))
            result = ReadUIntPixels(stream, width, height);
        else
            result = ReadInt32Pixels(stream, width, height);
    }
    else if (bitpix == -32) result = ReadFloatPixels(stream, width,
        height);
    else if (bitpix == -64) result = ReadDoublePixels(stream, width,
        height);
    else throw new NotSupportedException($"Formato BITPIX {bitpix} non
        supportato.");

    // Allineamento allo standard: salto del padding per completare il
        record di 2880 byte
    SkipPadding(stream, width, height, Math.Abs(bitpix) / 8);

    return result;
}
```

Figura 3.1. Implementazione del dispatcher ReadMatrix con supporto alla calibrazione BZERO e gestione del padding.

L'analisi della Figura 3.1 evidenzia come l'architettura software risolva l'ambiguità nativa dello standard FITS riguardo agli interi senza segno. Poiché il formato non prevede un valore `BITPIX` specifico per i tipi *unsigned* a 16 o 32 bit, il metodo `ReadMatrix` interroga preventivamente le keyword di calibrazione `BZERO` e `BSCALE`. Attraverso l'ausilio della funzione `IsUnsigned16`, il software identifica se il dataset rappresenta l'output grezzo di un sensore CCD astronomico, deviando il flusso verso il metodo ottimizzato `ReadUShortPixels`. Un aspetto fondamentale per la conformità allo standard è l'invocazione di `SkipPadding` al termine della lettura. Questa procedura assicura che il puntatore dello stream sia correttamente allineato all'inizio dell'HDU successivo, rispettando rigorosamente la struttura a record logici di 2880 byte descritta precedentemente.

Infine, si osservi che l'attuale implementazione non include il supporto per gli interi a 64 bit ($BITPIX = 64$). Questa scelta è motivata da precise ragioni di efficienza e finalità d'uso riscontrate durante la progettazione dell'architettura:

- **Complessità della rappresentazione Unsigned:** lo standard FITS definisce i dati a 64 bit nativamente come interi con segno. Per rappresentare valori senza segno, lo standard impone l'uso di un offset `BZERO` estremamente elevato, pari a 9223372036854775808 (ovvero 2^{63}), come indicato nei requisiti per la mappatura degli interi [1]. La gestione di costanti di tale magnitudine e le relative conversioni aritmetiche risulterebbe inutilmente onerosa in termini di complessità del codice e cicli di clock spesi per la normalizzazione del dato.
- **Requisiti operativi del software:** le logiche di analisi dell'applicativo (es. calcolo dei centroidi) operano quasi esclusivamente nel dominio dei numeri reali per garantire la massima accuratezza sub-pixel. Di conseguenza, poiché i dati grezzi interi necessiterebbero in ogni caso di una conversione in virgola mobile per poter essere processati dai motori matematici, il supporto diretto agli interi a 64 bit risulterebbe computazionalmente ridondante.
- **Rarità del formato e alternativa Double Precision:** nella pratica astronomica corrente, l'utilizzo di interi a 64 bit è limitato.

3.2 Strutture Dati Complesse e Compressione

3.2.1 Oltre l'Immagine Singola: MEF

Sebbene in origine il formato FITS sia nato per trasportare una singola matrice di dati per file, la sua evoluzione moderna lo ha portato ad operare come un contenitore modulare, noto come *MEF*. Questa architettura permette di aggregare in un unico file non solo l'immagine scientifica principale, ma anche una serie di dati accessori fondamentali per l'analisi, come mappe di calibrazione o ulteriori frammenti di dati. Tali estensioni garantiscono che tutte le informazioni necessarie rimangano indissolubilmente legate alla sorgente, facilitando l'archiviazione e la portabilità.

Per questo motivo, dal punto di vista implementativo, il servizio di lettura non può basarsi su un semplice marcatore di fine file (End Of File (EOF)) o su una lettura sequenziale indiscriminata. Invece, deve essere calcolato dinamicamente l'offset di memoria blocco interpretando i metadati dell'HDU corrente per determinare con precisione la fine dell'area dati e l'inizio del padding obbligatorio.

La dimensione complessiva occupata su disco dalla *Data Unit*, espressa come multiplo intero del record logico di 2880 byte, viene ricavata tramite la seguente relazione:

$$\text{DataSize} = \left\lceil \frac{|BITPIX| \times GCOUNT \times (PCOUNT + \prod_{i=1}^{n} NAXIS_i)}{8 \times 2880} \right\rceil \times 2880 \text{ byte} \quad (3.1)$$

Quindi, in questa formula si tiene conto dei seguenti parametri fondamentali definiti dallo standard per la gestione di strutture dati raggruppate o estensioni con parametri extra:

- **PCOUNT (Parameter Count):** indica il numero di parametri che seguono l'array principale.

- **GCOUNT (Group Count):** specifica il numero di gruppi di dati identici presenti nell'estensione.
- $\prod_{i=1}^{n} NAXIS_i$**:** rappresenta il prodotto delle dimensioni di tutti gli assi definiti, calcolato nel software tramite il ciclo di scansione delle keyword `NAXISn`.
- **Operatore di Ceiling ($\lceil\rceil$):** assicura che il numero di blocchi sia arrotondato per eccesso all'intero più vicino, garantendo l'allineamento al limite dei 2880 byte richiesto dal formato.

L'applicazione corretta di questa formula permette al software di riposizionare il puntatore dello stream esattamente all'inizio dell'HDU successivo, evitando errori di parsing nelle letture multi-estensione.

3.2.2 Tabelle Binarie e Gestione Dinamica (Heap)

Oltre alle immagini, l'analisi astrometrica produce grandi quantità di dati strutturati: liste di coordinate, effemeridi orbitali o log della strumentazione. Per gestire queste informazioni, lo standard definisce due tipologie di tabelle:

ASCII Tables (XTENSION = 'TABLE '): Memorizzano i dati in formato testuale leggibile. Pur essendo utili per piccoli dataset ispezionabili visivamente, risultano inefficienti per volumi massivi di dati a causa dell'overhead richiesto per la conversione da stringa a valore numerico in fase di lettura [17].

Binary Tables (XTENSION = 'BINTABLE'): Rappresentano lo standard per l'archiviazione di dati complessi. I dati sono organizzati in righe e colonne e memorizzati direttamente in formato binario (Big-Endian). Questa struttura permette un accesso estremamente rapido tramite memory mapping, trattando le colonne come array nativi. Ogni colonna è descritta nell'header dalle keyword `TFORMn`, che ne specificano il tipo (es. `1E` per un float a 32 bit, `1D` per un double a 64 bit) [18].

Più nello specifico, una delle caratteristiche più sofisticate delle Binary Tables è il supporto per gli *Array a Lunghezza Variabile*. Infatti, in una tabella standard ogni riga ha una dimensione fissa, il che facilita l'accesso diretto. Tuttavia, alcuni dati scientifici sono intrinsecamente irregolari, e questa implementazione risulterebbe essere una forte limitazione.

Per superare questo limite senza compromettere la rigidità della struttura tabulare, lo standard FITS implementa un'area di memoria supplementare denominata Heap, posizionata immediatamente dopo la fine della tabella principale. L'accesso a questi dati a lunghezza variabile è mediato da un sistema di descrittori: la singola cella della tabella, infatti, non ospita direttamente il dato reale, bensì un puntatore logico formato da due valori interi. Il primo intero quantifica il numero esatto di elementi presenti in quella specifica riga, mentre il secondo fornisce l'offset in byte a partire dall'inizio dello Heap, indicando l'indirizzo in cui risiedono le informazioni effettive.

Sebbene questa architettura imponga al software una logica di decodifica e allocazione dinamica decisamente più raffinata, essa conferisce al formato una straordinaria flessibilità, permettendogli di incapsulare strutture dati eterogenee che altrimenti richiederebbero formati molto più pesanti o meno standardizzati.

3.2.3 La "Tiled Image Compression Convention"

La descrizione dettagliata delle tabelle e dei record logici presentata nelle sezioni precedenti non deve essere considerata un approfondimento fine a se stesso. Sebbene il software sia progettato principalmente per la gestione e la visualizzazione delle immagini astronomiche, la comprensione di queste strutture è propedeutica all'analisi delle tecniche di compressione e decompressione previste dallo standard.

Infatti, l'enorme volume di dati associato ai moderni dati astronomici ha reso la gestione dello spazio di archiviazione una sfida critica. Tuttavia, l'uso di algoritmi di compressione generici

introduce un grosso limite operativo: l'impossibilità di eseguire un accesso casuale [2]. Per leggere anche solo un singolo pixel in un file compresso sequenzialmente, il software sarebbe costretto a decomprimere l'intero dataset, con un enorme dispendio di risorse computazionali.

Per risolvere questa criticità, lo standard FITS ha introdotto la *Tiled Image Compression Convention*. In questa configurazione, il file compresso non è più memorizzato come un'estensione immagine standard, ma viene trasformato in una Binary Table altamente ottimizzata. Il processo si articola in tre fasi:

1. **Tassellazione (Tiling):** L'immagine originale di dimensioni $N \times M$ viene suddivisa in una griglia di mattonelle rettangolari indipendenti (ad esempio, blocchi di 100×100 pixel).
2. **Compressione Locale:** Ogni tassello viene compresso singolarmente utilizzando algoritmi specifici. I più diffusi sono gli algoritmi Rice, GZIP e H-Compress.
3. **Archiviazione:** Il flusso di byte compresso di ogni tassello viene salvato come una riga all'interno di una colonna a lunghezza variabile (`COMPRESSED_DATA`) di una Binary Table. Keyword dedicate come `ZSCALE` e `ZZERO` conservano i parametri necessari per la ricostruzione del dato originale (lossless) o per la quantizzazione (lossy).

Per quanto riguarda l'implementazione dello standard Tile Compression nel software, il codice si basa su una netta separazione delle responsabilità tra le fasi di scrittura e lettura, gestite rispettivamente dalle classi `FitsCompression` e `FitsDecompression`.

Il compito principale di `FitsCompression` è quello di orchestrare la traduzione bidirezionale dei metadati. Attraverso il metodo `CreateCompressedHeader`, il servizio "congela" le informazioni dell'immagine originale trasformandole in Z-Keywords. Ad esempio, le dimensioni reali dell'immagine vengono rinominate in `ZNAXIS1` e `ZNAXIS2`, mentre le keyword standard `NAXISn` dell'header compresso vengono impostate per descrivere la geometria della tabella binaria che ospiterà i dati.

Un aspetto cruciale in questa fase è la gestione della quantizzazione statistica per i dati in virgola mobile. Poiché algoritmi come Rice sono estremamente efficienti sugli interi ma non sui float, il software implementa la logica di fpack per convertire i pixel in interi a 32 bit, calcolando per ogni tassello i parametri di riscalamento `ZSCALE` e `ZZERO`. Questi valori vengono poi memorizzati in colonne dedicate della tabella, garantendo che l'operazione sia reversibile con la massima precisione.

Specularmente, `FitsDecompression` agisce come un motore di ricostruzione che deve ricomporre il puzzle dei tasselli compressi. La logica di questo servizio non è sequenziale rispetto ai pixel, ma rispetto alle righe della tabella: ogni riga viene letta per estrarre il descrittore dello Heap, che contiene l'indirizzo e la lunghezza del payload binario compresso. La complessità maggiore in questa fase risiede nel calcolo della geometria dei tasselli. Il software deve determinare le coordinate esatte (tx, ty) di ogni tile all'interno della matrice di destinazione. Una volta ottenuto il blocco decompresso tramite i codec Rice o GZIP, i dati vengono riportati alla loro scala originale (se quantizzati) e copiati nella posizione corretta della matrice finale tramite operazioni di `Buffer.BlockCopy` ad alte prestazioni.

3.2.4 Algoritmi di Compressione: Rice e GZIP

La selezione degli algoritmi di compressione implementati nel modulo è stata guidata dalla precisa necessità di garantire la massima interoperabilità con lo standard *fpack* e di gestire in modo efficiente la forte eterogeneità dei dati astronomici. Per questo motivo, lo sviluppo si è focalizzato sui due algoritmi cardine definiti dalla *Tiled Image Convention*: Rice e GZIP.

Sebbene GZIP (basato sull'algoritmo DEFLATE) rappresenti lo standard industriale per l'archiviazione di dati generici, la sua applicazione diretta alle immagini grezze dei sensori CCD risulta spesso inefficace. Tali acquisizioni sono infatti dominate dal rumore di Poisson e dal rumore di lettura elettronico, componenti stocastiche intrinsecamente prive di ridondanza strutturale. Poiché GZIP opera ricercando pattern ripetitivi attraverso la costruzione di un dizionario LZ77,

l'elevata entropia del rumore astronomico impedisce la creazione di riferimenti validi. Questo limite conduce frequentemente a rapporti di compressione inferiori alle aspettative, anche a causa dell'*overhead* introdotto dal dizionario stesso.

Al contrario, per i dati interi (16 o 32 bit), lo standard [2] privilegia l'algoritmo Rice (Golomb-Rice coding). Essendo un codificatore entropico puro che opera sulle differenze tra pixel adiacenti, Rice sfrutta l'elevata correlazione spaziale delle sorgenti astronomiche: i gradienti tra pixel vicini sono dolci e le differenze numeriche piccole, permettendo una codifica estremamente compatta anche in presenza di rumore.

Tuttavia, per quanto riguarda l'applicazione di questi algoritmi ai dati in virgola mobile (32/64 bit) c'è da fare una precisazione. Poiché Rice è matematicamente definito per i numeri interi, la compressione di valori *floating point* richiede una fase preliminare di quantizzazione. Questo processo introduce tecnicamente una perdita di informazione (*lossy compression*), mappando i valori continui su livelli discreti. Tuttavia, come documentato da White et al. [19], tale perdita è controllata per essere statisticamente trascurabile: impostando il fattore di quantizzazione al di sotto del rumore di fondo intrinseco (σ_{noise}), si scartano esclusivamente le fluttuazioni casuali dei bit meno significativi, preservando intatta la precisione fotometrica.

In questo scenario, GZIP assume un ruolo fondamentale: a differenza di Rice, esso può operare direttamente sul flusso di byte dei float senza quantizzazione. È proprio per questo motivo che la strategia di compressione con GZIP viene mantenuta come alternativa necessaria per garantire una compressione puramente *lossless* nei casi in cui l'alterazione del dato, seppur minima, non sia accettabile.

Infine, l'implementazione dell'algoritmo H-Compress (basato su trasformata wavelet di Haar) è stata esclusa sulla base delle limitazioni evidenziate nello standard stesso [2]. Sebbene teoricamente reversibile, l'algoritmo risulta "generalmente inefficace" per la compressione lossless di immagini rumorose. La sua efficacia emerge solo in modalità lossy, dove però tende a introdurre artefatti geometrici (*blockiness*) che richiedono un post-processing di smoothing in fase di decompressione. Tale manipolazione estetica del segnale è stata ritenuta incompatibile con gli obiettivi di rigore scientifico perseguiti da questa libreria.

L'implementazione del codec Rice si è rivelata essere una sfida significativa dello sviluppo. Infatti, analizzando le librerie presenti online è emersa la mancanza di librerie aggiornate e attualmente supportate capaci di gestire questo standard di compressione in modo efficiente per quanto riguarda le immagini FITS. Di conseguenza, è stato necessario sviluppare da zero il motore di codifica, effettuando un porting manuale basato direttamente sui sorgenti C originali della National Aeronautics and Space Administration (NASA) della libreria `cfitsio`.

Inizialmente era stata presa in considerazione l'idea di utilizzare direttamente il codice nativo C (tramite P/Invoke), ma questa strada è stata scartata. Infatti, questa soluzione avrebbe introdotto dipendenze specifiche per ogni sistema operativo, compromettendo la portabilità del progetto. La scelta di riscrivere tutto in C# ha permesso invece di sfruttare appieno la natura multipiattaforma senza dover compilare driver nativi o gestire configurazioni complesse.

Il risultato è la classe `RiceCodec`, che implementa la logica di codifica a livello di singolo bit per assicurare una compatibilità *bit-exact* con lo standard fpack. Poiché i codici Rice hanno lunghezza variabile e non sono allineati ai byte, il codec accumula i bit in un buffer interno e li scrive nello stream solo al completamento del byte:

```
// KomaLab.Services.Fits.IO.RiceCodec
// Gestione della codifica bit-level per compatibilita' fpack (porting cfitsio)
public static byte[] Encode(int[] a, int nblock = 32)
{
    using var ms = new MemoryStream();
    var buf = new OutputBuffer(ms);
    // ... calcolo differenze e parametro 'fs' ...

    for (int j = 0; j < thisblock; j++)
    {
        uint v = diffs[j];
        int top = (int)(v >> fs); // Parte quoziente (codifica unaria)

        // 1. Codifica Unaria (Leading Zeros): scrive 'top' zeri
        // Questa logica bit-exact e' necessaria per il decoder NASA
        for(int k=0; k < top; k++) buf.OutputNBits(0, 1);

        // 2. Terminatore: scrive un 1
        buf.OutputNBits(1, 1);

        // 3. Codifica Binaria del resto (se fs > 0)
        if (fs > 0) buf.OutputNBits(v & (uint)((1 << fs) - 1), fs);
    }
    buf.Done(); // Flush del padding finale
    return ms.ToArray();
}

// Helper per la scrittura MSB-first non allineata ai byte
private class OutputBuffer
{
    public void OutputNBits(uint bits, int n)
    {
        for (int i = n - 1; i >= 0; i--)
        {
            uint bit = (bits >> i) & 1;
            if (bit != 0) _currentByte |= (byte)(1 << (7 -
                _bitsFilled));

            if (++_bitsFilled == 8)
            {
                _stream.WriteByte(_currentByte);
                _currentByte = 0;
                _bitsFilled = 0;
            }
        }
    }
}
```

Figura 3.2. Core logic del codec Rice: gestione bit-level e codifica entropica.

3.3 Interfaccia Utente: Importazione ed Esportazione

L'architettura software descritta nelle sezioni precedenti, per quanto robusta, necessita di un livello di interazione che astragga la complessità dei formati FITS e dei processi di compressione. A tal fine, sono state realizzate tre finestre distinte con le relative interfacce per guidare l'astronomo attraverso tre fasi critiche: l'acquisizione e pre-elaborazione del dato (*Import*), la salvaguardia del risultato scientifico (*Scientific Export*) e la produzione di materiali per la divulgazione (*Media Export*).

3.3.1 Importazione e Pipeline di Calibrazione

La finestra di dialogo per l'importazione (visibile in Figura 3.3) rappresenta il punto di ingresso dei dati nel sistema ed è accessibile navigando nel menù *File* → *Aggiungi Nodo*. Progettata per gestire il caricamento *batch*, permette all'utente di selezionare più di una singola immagine per volta in modo da rendere più agevole il caricamento di grossi dataset di immagini. Il sistema analizza in tempo reale gli header dei file selezionati: nel caso di file compressi o MEF, il software gestisce automaticamente l'estrazione delle singole HDUs, presentando all'utente una lista di immagini importabili.

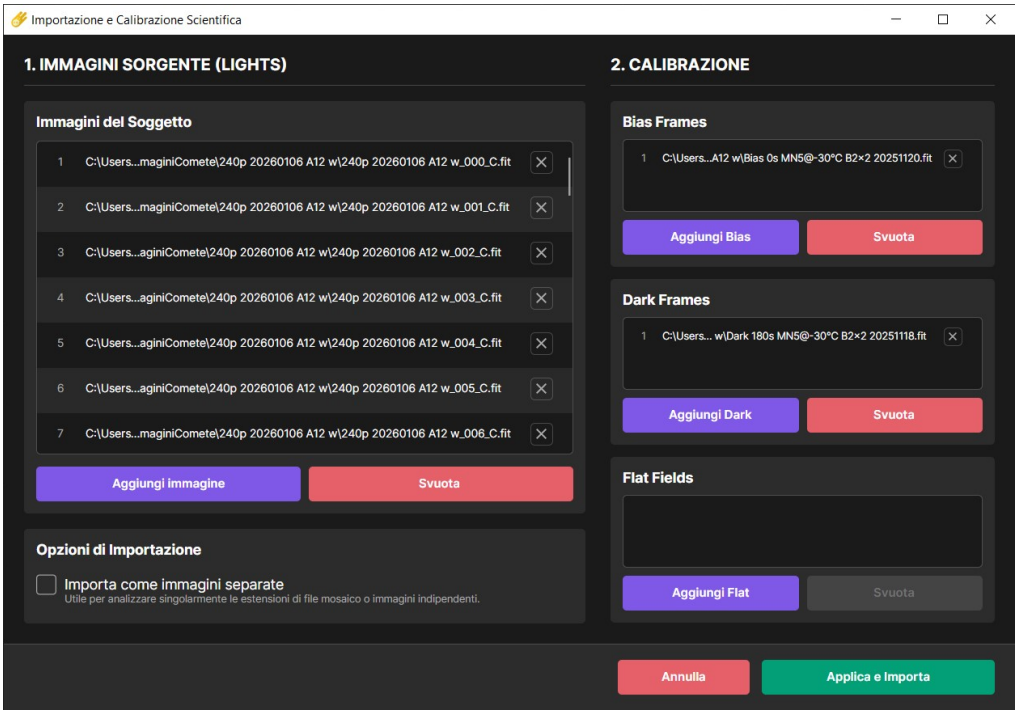

Figura 3.3. Interfaccia di importazione con selezione multipla e gestione dei Master Frames.

Inoltre, in questa finestra l'utente può decidere se caricare la sequenza di immagini in un singolo nodo multiplo (utile per serie temporali della stessa cometa) o distribuirla su nodi separati.

La Necessità della Calibrazione Prima che il dato grezzo venga convertito nelle strutture in memoria (`FitsDataPackage`), l'UI offre la possibilità di applicare una calibrazione preliminare. Le immagini astronomiche *raw*, infatti, sono affette da imperfezioni strumentali sistematiche che devono essere sottratte per ottenere una misura scientificamente valida. La UI permette di caricare tre tipologie di *Master Frames*:

- **Bias Frames**: utilizzati per campionare e rimuovere il rumore elettronico di lettura intrinseco del sensore, operando su pattern catturati a tempo di posa nullo.

- **Dark Frames**: essenziali per sottrarre la corrente di buio, ovvero il rumore termico generato dal sensore in proporzione alla temperatura e alla durata dell'esposizione.
- **Flat Fields**: impiegati per compensare la vignettatura ottica e le disomogeneità nella sensibilità dei singoli pixel tramite la divisione dell'immagine per una mappa di luce uniforme, preventivamente normalizzata rispetto al suo valore medio.

L'implementazione degli algoritmi relativi a tale processo è demandata al `CalibrationEngine`, la quale sfrutta operazioni di algebra lineare su matrici ad alta precisione fornite dalla libreria OpenCV. Al fine di preservare la linearità del segnale e scongiurare errori di troncamento durante le computazioni, il motore di calcolo converte preventivamente le immagini in un formato a virgola mobile (*floating-point*) a 32 o 64 bit, coerentemente con la profondità del dato grezzo originario.

Il flusso di calcolo (*pipeline*) segue una sequenza rigorosa, indispensabile per garantire la piena validità scientifica del risultato:

1. **Sottrazione del rumore additivo:** il motore esegue la sottrazione matriciale del *Master Dark* (o, in sua assenza, del *Master Bias*) dal *light frame* grezzo, eliminando così la componente termica e il rumore di lettura strumentale.
2. **Preparazione della risposta ottica:** prima di procedere alla correzione del campo, il *Master Flat* viene a sua volta calibrato sottraendovi il *Bias* (o il *Dark Flat*), isolando in questo modo la pura disomogeneità dell'illuminazione sensore-ottica. Successivamente, la matrice del *Flat* calibrato viene normalizzata dividendola per la propria media aritmetica globale.
3. **Divisione di campo (*Flat-Fielding*):** l'immagine empirica, precedentemente depurata dal rumore additivo, viene divisa punto-punto per la matrice del *Flat* normalizzato. In questo frangente, l'algoritmo introduce una soglia di tolleranza infinitesima al denominatore (*epsilon*), prevenendo instabilità numeriche o divisioni per zero in corrispondenza di pixel anomali o zone periferiche non esposte.
4. **Finalizzazione e *Clipping*:** il processo si chiude con un'operazione di troncamento (*clipping*), che forza a zero eventuali valori numerici negativi scaturiti dalle sottrazioni, ripristinando la rigorosa coerenza fisica del dominio della radiazione.

L'applicazione sistematica di questi *master frame* già in fase di importazione assicura che l'intero flusso di lavoro successivo operi esclusivamente su dataset radiometricamente calibrati e linearmente corretti.

3.3.2 Ispezione e Modifica dei Metadati (FITS Header Editor)

Tra la fase di acquisizione e le successive elaborazioni scientifiche o esportazioni, si colloca una funzionalità di vitale importanza per il controllo qualità del dato: l'interfaccia di visualizzazione e modifica dell'header FITS.

Molto spesso, le immagini grezze acquisite da setup amatoriali o da strumentazione non perfettamente configurata presentano metadati mancanti o errati. Tali lacune inibiscono i successivi algoritmi di risoluzione astrometrica (*Plate Solving*) o di calcolo delle effemeridi. Per ovviare a questo problema, Kometra [3] mette a disposizione un avanzato *Header Editor* integrato, accessibile navigando nel menù alla voce *Modifica* → *Editor Header / Metadati* dopo aver selezionato un nodo.

Dal punto di vista dell'interazione utente, l'interfaccia dell'editor in questione è suddivisa in due macro-colonne: una dedicata alla diagnostica e l'altra all'editing puro.

La colonna di sinistra espone il nome dell'immagine corrente e fornisce un riassunto diagnostica preliminare dei metadati. Infatti, il motore software valuta in tempo reale la presenza e la validità sintattica di cinque macro-categorie di informazioni, restituendo un *feedback* visivo immediato tramite indicatori colorati (indicatore rosso per dati mancanti, verde per dati corretti). I cinque ambiti chiave valutati dal sistema sono:

- **Riferimento Temporale:** verifica della presenza della keyword `DATE-OBS`, parametro imprescindibile per i calcoli di meccanica celeste e per la determinazione orbitale.
- **Geolocalizzazione:** controllo delle coordinate di latitudine e longitudine del sito osservativo (`SITELAT`, `SITELONG`), necessarie al software per calcolare accuratamente la parallasse topocentrica.
- **Configurazione Ottica:** Dati fisici del setup di ripresa, ovvero lunghezza focale (`FOCALLEN`) e dimensione fisica dei pixel (`PIXSIZE`), determinanti per il calcolo del campionamento spaziale.
- **Coordinate di Puntamento:** Coordinate equatoriali stimate del centro dell'immagine (`RA`, `DEC`), utilizzate come seed per gli algoritmi di astrometria.
- **Soluzione Astrometrica:** accertamento dell'avvenuta integrazione delle matrici di trasformazione spaziale WCS all'interno dell'intestazione, a certificazione del corretto completamento della procedura di *Plate Solving*.

Questo sistema di *User Experience* proattiva permette all'astronomo di capire con una singola occhiata se l'immagine è scientificamente pronta o se necessita di correzioni manuali.

Invece, nella colonna di destra dell'interfaccia viene estratta la complessa architettura a stringhe di 80 caratteri (*card images*) imposta dallo standard, decodificando l'intera lista dei metadati in una leggibile struttura tabellare. Per facilitare la navigazione all'interno di header, che possono contenere centinaia di record, è integrata una barra di ricerca testuale reattiva. L'operatore può ispezionare e sovrascrivere i valori esistenti, o utilizzare gli appositi comandi per aggiungere nuove righe alla tabella.

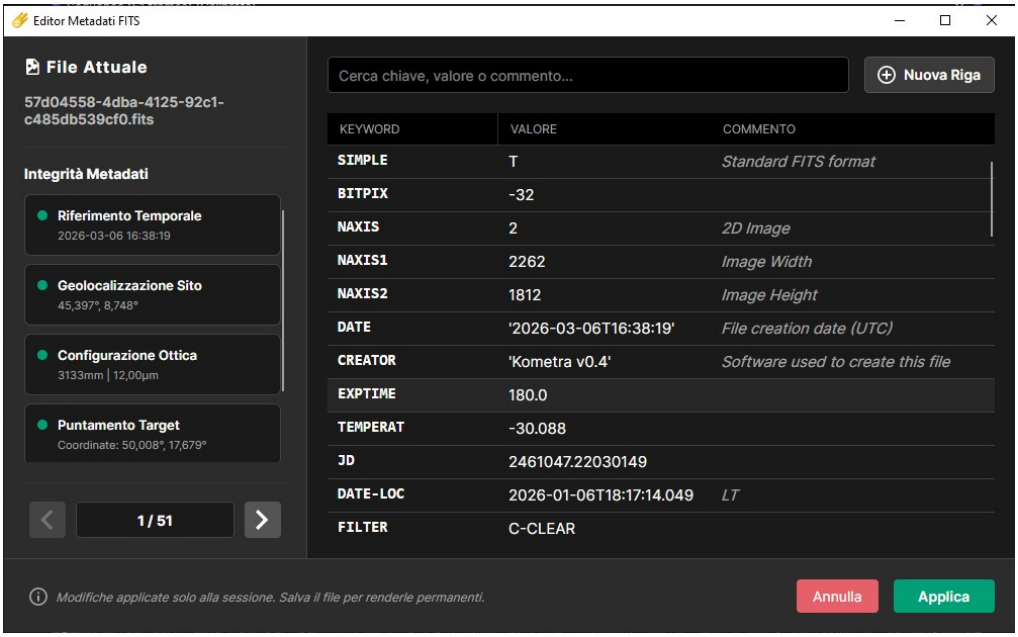

Figura 3.4. Interfaccia del FITS Header Editor. A sinistra sono riportati gli indicatori di validazione dei parametri fondamentali, mentre a destra si sviluppa la vista tabellare completa, dotata di strumenti per la ricerca e la modifica dei record logici.

3.3.3 Salvataggio ed Esportazione Immagini

La fase di esportazione è accessibile una volta selezionato un nodo all'interno dello spazio di lavoro. L'utente può richiamare questa finestra utilizzando il percorso *File* → *Salva Immagine* dal menù principale.

Una volta aperta la finestra, l'interfaccia è divisa logicamente in tre aree: a sinistra, una *checklist* permette di selezionare puntualmente quali immagini della serie salvare; al centro è possibile visualizzare un'immagine per volta, tutte quelle presenti nella lista; a destra, il pannello di configurazione definisce il formato e la destinazione (Figura 3.5). Il sistema gestisce automaticamente la nomenclatura dei file tramite un suffisso incrementale numerico nel caso di salvataggi multipli.

Relativamente ai formati di esportazione scientifica, l'interfaccia espone all'utente le seguenti opzioni:

- **FITS Standard**: salvataggio del dataset originale in formato puro non compresso, garantendo l'integrità assoluta dei dati grezzi senza alcuna manipolazione binaria.
- **FITS Compresso**: implementazione della *Tiled Image Compression* con possibilità di scelta tra l'algoritmo
- **FITS Compresso:** applicazione della *Tiled Image Compression*. Il sistema consente di selezionare l'algoritmo **Rice** (ottimizzato per massimizzare l'efficienza sui dati interi) oppure **GZIP** (impiegato per preservare l'esatta precisione dei dati in virgola mobile).

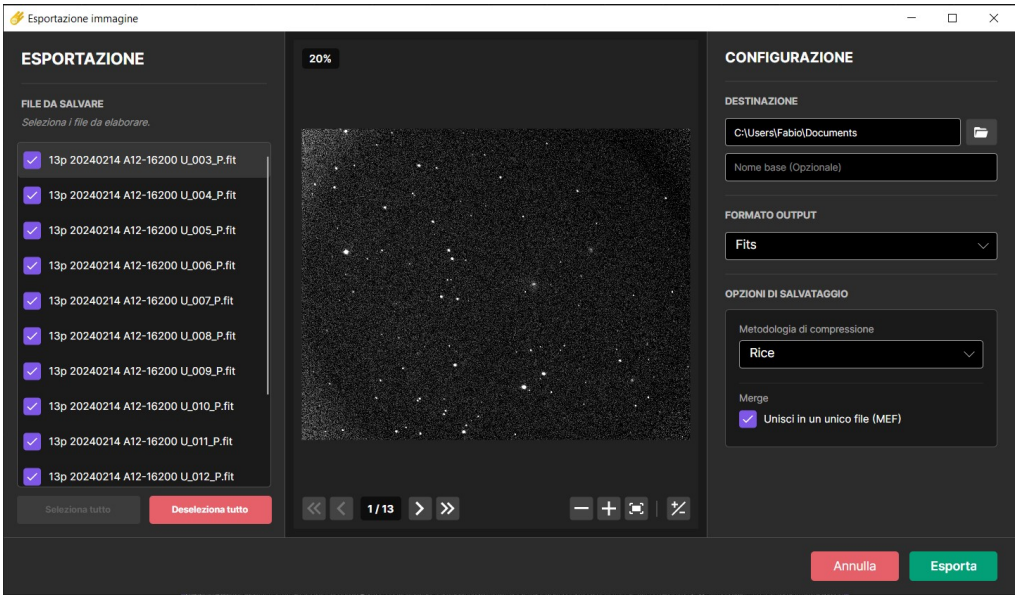

Figura 3.5. Pannello dedicato all'esportazione: sono visibili i controlli per la selezione del formato (FITS Standard/Compresso) e per l'opzione MEF.

In entrambi i casi è poi possibile selezionare l'opzione di impacchettamento di tutte le immagini selezionate in un unico file contenitore MEF o mantenerle come file separati.

Oltre ai formati scientifici, è supportata l'esportazione in formati grafici standard (PNG, JPEG). Poiché questi formati operano a 8 bit per canale (contro i 16/32 bit del dato astronomico), il software esegue un *tone mapping* non lineare. Le soglie di taglio per il bianco e il nero (*White/Black Points*) possono essere determinate automaticamente o regolate manualmente dall'utente tramite gli slider che compariranno nella colonna di destra o tramite l'anteprima centrale, permettendo di evidenziare dettagli della chioma cometaria altrimenti invisibili.

3.3.4 Esportazione Video

Una funzionalità aggiuntiva, implementata specificamente per scopi divulgativi, è la possibilità di esportare brevi video a partire da una sequenza di immagini. Questa interfaccia viene aperta selezionando un nodo contenente una serie temporale e accedendo al menù *File* → *Salva Video*.

L'analisi cometaria si basa spesso su lunghe serie temporali che mostrano il moto dell'oggetto rispetto al campo stellato fisso; l'export video permette di condensare queste osservazioni in brevi filmati (Figura 3.6).

Il pannello di configurazione espone parametri avanzati per il controllo della codifica e del formato che si desiderano utilizzare:

- **Formato contenitore:** supporto per i principali standard di settore, quali `.MPEG-4 Part 14 (MP4)`, `.Audio Video Interleave (AVI)` e `.Matroska Video (MKV)`, garantendo la massima flessibilità nella distribuzione dei contenuti generati.
- **Codec:** possibilità di selezione tra codec *legacy* per assicurare la compatibilità con sistemi datati (`MJPG, Xvid`) e standard moderni ad alta efficienza (`H.264, H.265/HEVC`), ottimizzati per preservare l'integrità visiva pur riducendo sensibilmente il *bitrate* complessivo.
- **Parametri temporali e geometrici:** controllo granulare della frequenza dei fotogrammi (*frame rate*), finalizzato alla modulazione della dinamica temporale dell'evento (es. *time-lapse*), e applicazione di algoritmi di riscalamento (*downscaling*) per convertire le risoluzioni native elevate (4K/8K) dei sensori astronomici in formati più agili per la fruizione web, come il 1080p.

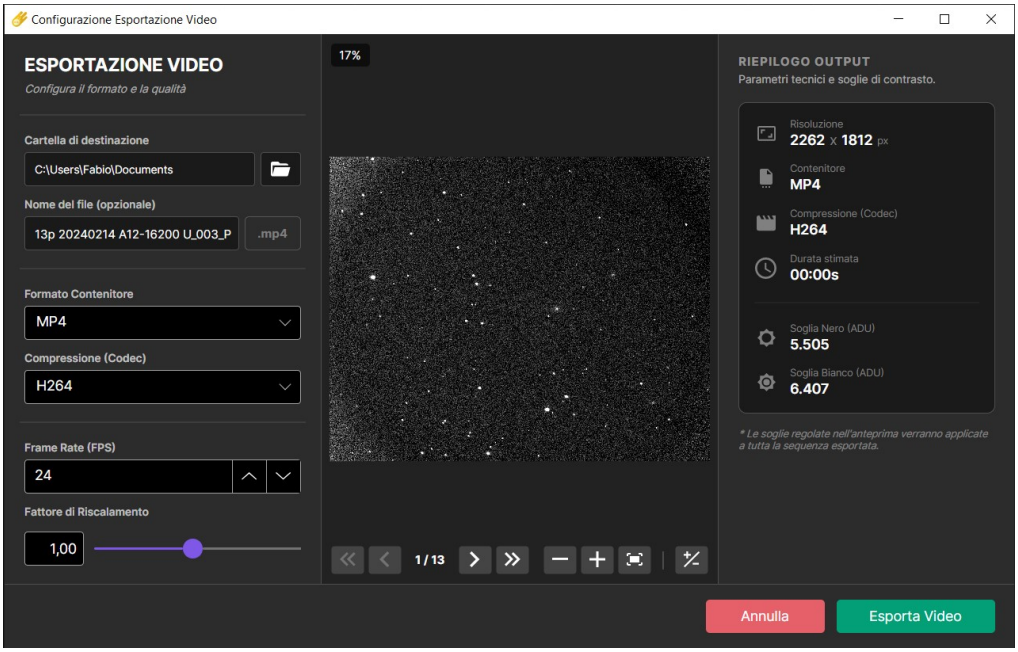

Figura 3.6. Interfaccia per la generazione di timelapse con controlli per codec e frame rate.

Anche in questo caso, il risultato visivo è determinato dal valore assegnato alle soglie di visualizzazione (livelli) dell'immagine di riferimento, garantendo in questo modo che il video finale rispecchi fedelmente l'elaborazione effettuata dall'astronomo.

Capitolo 4

Rilevamento del Nucleo e Allineamento Astrometrico

Dopo aver definito le metodologie di acquisizione e gestione del dato grezzo, il presente capitolo affronta una delle sfide computazionali e astrofisiche più complesse nell'elaborazione di immagini cometarie: l'identificazione precisa del nucleo e il successivo allineamento spaziale dei fotogrammi. L'analisi partirà da una disamina della morfologia e della dinamica fisica delle comete. Successivamente, verranno esplorati in dettaglio gli algoritmi matematici sviluppati per il rilevamento sub-pixel dell'optocentro, per concludere con l'illustrazione dell'infrastruttura astrometrica e dell'architettura software che permettono di ancorare tali misurazioni al sistema di riferimento celeste.

4.1 Natura, Morfologia e Dinamica Cometaria

Prima di entrare nello specifico delle tecniche e delle problematiche relative all'allineamento delle immagini sul centro cometario, è necessario fare un passo indietro per comprendere cosa siano effettivamente le comete, come siano strutturate e quali siano le loro principali caratteristiche. Infatti, avere ben chiara la natura e la morfologia di questi oggetti permette di comprendere meglio quali siano le sfide presenti nell'individuazione e nel tracciamento del loro nucleo.

4.1.1 Origine e Serbatoi: Nube di Oort e Fascia di Kuiper

Le comete sono corpi minori riconosciuti ad oggi tra i depositi più antichi e inalterati di materiale della nebulosa solare primordiale, formatasi oltre 4,6 miliardi di anni fa. A differenza di asteroridi e meteoriti, la loro composizione include ghiacci volatili, polveri refrattarie e complessi composti organici e funge da archivio delle condizioni chimico-fisiche che hanno preceduto la formazione dei pianeti [20]. Dopo la loro formazione, le comete si sono distribuite in due regioni principali del sistema solare esterno:

- La **Fascia di Kuiper**: È una grande regione composta da corpi ghiacciati e situata ben oltre l'orbita di Nettuno. Il suo bordo interno inizia a circa 30 Unità Astronomicas (UAs) dal Sole, mentre la regione principale termina a circa 50 UAs. Questa regione, che include anche Plutone, è la fonte delle comete a corto periodo, le quali compiono orbite brevi della durata di 20 anni o meno [21].
- La **Nube di Oort**: È un guscio sferico estremamente distante e vasto di corpi ghiacciati che circonda il sistema solare. Si estende per una distanza che va da circa un quarto fino a metà strada tra il nostro Sole e la stella successiva, con un bordo interno situato tra le 2.000 e le 5.000 UAs e un bordo esterno tra le 10.000 e le 100.000 UAs dal Sole. Questa nube spiega l'origine delle comete a lungo periodo (con periodi orbitali di migliaia di anni)

e si stima che al suo interno possano esserci centinaia di miliardi, o persino trilioni, di corpi ghiacciati [22].

L'astronomo Jan Oort ipotizzò che queste comete siano state mosse gravitazionalmente verso queste zone a causa dell'influenza dei pianeti giganti o dalla gravità della galassia stessa [20].

4.1.2 Evoluzione Storica e Modelli Teorici della Struttura Nucleare

Negli anni '50 vi fu un intenso dibattito sulla natura fisica delle comete, innescato dalla teoria del "banco di sabbia" (*sandbank model*) proposta da Ray Lyttleton nel 1948. Lyttleton ipotizzava che le comete fossero il risultato della coalescenza di flussi di polvere catturati gravitazionalmente dal Sole durante il suo passaggio attraverso nubi di polvere interstellare. Questo modello, tuttavia, presentava lacune critiche nella spiegazione della sopravvivenza dei nuclei (che non avrebbero resistito a numerosi passaggi al perielio) e nell'interpretazione delle anomalie orbitali non-newtoniane [23].

Nel 1950, l'astronomo Fred Whipple propose un modello rivoluzionario in contrasto con quello di Lyttleton, descritto come "conglomerato ghiacciato" (*icy conglomerate*) e noto popolarmente come l'ipotesi della "palla di neve sporca". Whipple postulò che il nucleo fosse un corpo solido e discreto, una miscela coerente di ghiacci volatili (principalmente acqua, monossido di carbonio e anidride carbonica) e polveri meteoritiche. L'intuizione fondamentale di Whipple fu l'applicazione di questo modello per spiegare le forze non gravitazionali: il processo di sublimazione asimmetrica, influenzato dalla rotazione del nucleo, genera una spinta reattiva (effetto jet) che altera la traiettoria orbitale. Poiché il nucleo impiega del tempo per riscaldarsi (inerzia termica), il picco massimo di emissione dei gas non avviene esattamente nel punto più vicino al Sole, ma si verifica con un leggero ritardo. Questi getti di gas agiscono come dei veri e propri propulsori: a seconda del senso di rotazione della cometa, la spinta generata può farla accelerare o frenare lungo la sua orbita. Questa dinamica ha finalmente permesso di spiegare i movimenti anomali osservati nella cometa 2P/Encke [23].

Il dibattito scientifico tra il modello del banco di sabbia e quello del conglomerato ghiacciato trovò una risoluzione definitiva solo nel 1986 con la missione Giotto dell'European Space Agency (ESA). Le immagini ravvicinate del nucleo della cometa 1P/Halley confermarono l'esistenza di un corpo solido, irregolare e scuro, validando l'essenza della teoria di Whipple pur introducendo nuove complessità riguardo alla morfologia superficiale e alla localizzazione dell'attività [23].

Questo modello però non si adatta perfettamente a tutte le comete e missioni successive, in particolare Deep Impact e i risultati definitivi della sonda Rosetta sulla cometa 67P/Churyumov-Gerasimenko, hanno dimostrato che in alcuni casi la componente dominante del nucleo non è il ghiaccio d'acqua, preferendo al modello della palla di neve sporca quello della "palla di sporco ghiacciata" (*icy dirtball*). In questa nuova e moderna visione, il nucleo cometario è descritto come un aggregato di polvere estremamente poroso (con vuoti fino all'80%) e debolmente legato, in cui la massa della frazione refrattaria domina nettamente su quella volatile [24].

4.1.3 Anatomia della Cometa: Dal Nucleo alle Code

Strutturalmente, una cometa attiva si articola in tre componenti fondamentali (si veda la Figura 4.1): il nucleo solido centrale, la chioma atmosferica circostante e le code che si estendono nello spazio. Ciascuna di esse è caratterizzata da specifiche proprietà fisiche e dinamiche, la cui comprensione risulta cruciale per contestualizzare le intrinseche difficoltà osservative di tali oggetti.

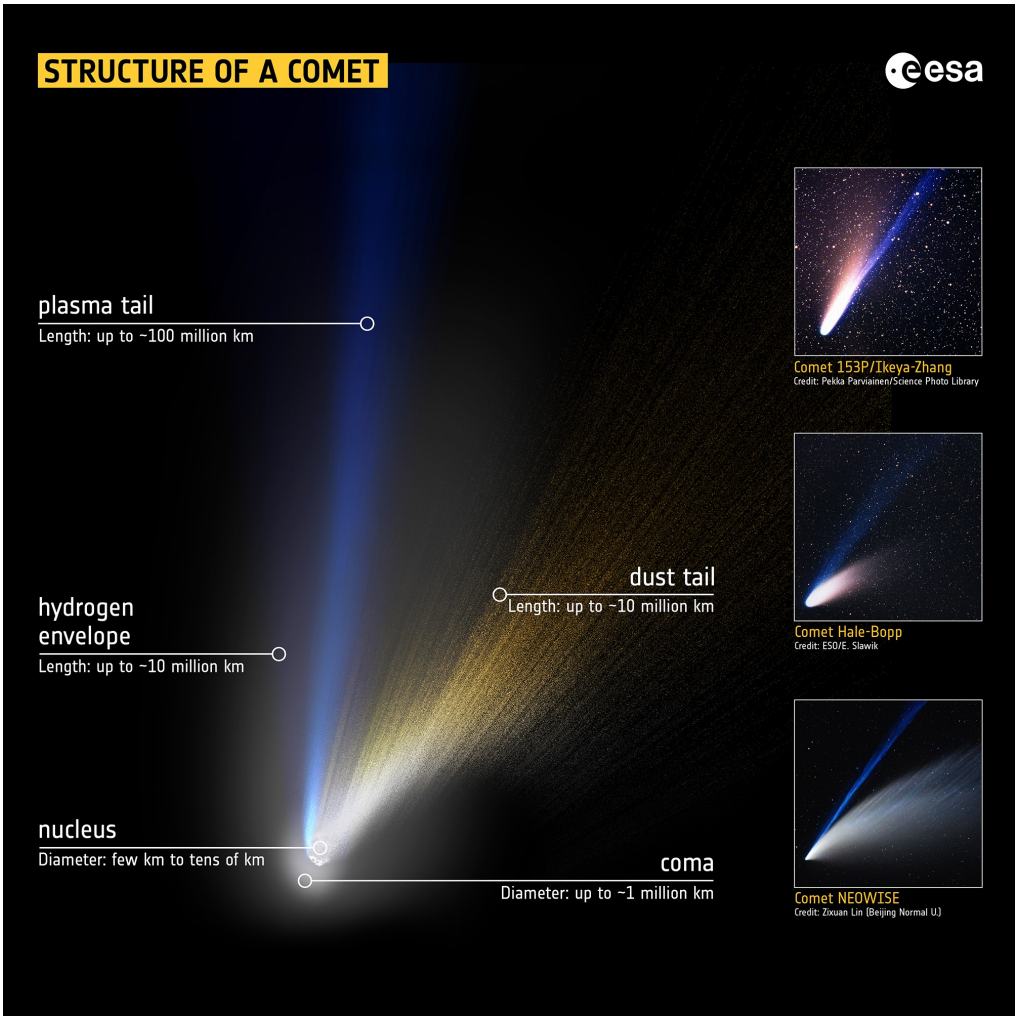

Figura 4.1. Struttura e anatomia di una cometa. Sono visibili il nucleo, la chioma, la coda di polveri e la coda di plasma (ioni), oltre all'involucro di idrogeno invisibile a occhio nudo. Crediti: ESA (Work performed by ATG under contract to ESA), licenza CC BY-SA 3.0 IGO [25].

Il Nucleo: Il Motore Dinamico Il nucleo costituisce l'unica componente solida della cometa, con dimensioni che variano tipicamente da poche centinaia di metri a decine di chilometri. Sebbene rappresenti il "cuore" dinamico dell'oggetto, le analisi fisiche ne rivelano un'estrema fragilità strutturale. Le misurazioni *in situ* effettuate dalla missione Rosetta sulla cometa 67P/Churyumov-Gerasimenko hanno fornito dati di precisione senza precedenti, identificando una densità di 533 ± 6 kg/m^3, valore pari a circa la metà di quello dell'acqua liquida. Tale dato, associato a una porosità stimata tra il 72% e il 74%, suggerisce una struttura interna simile a un "mucchio di macerie" (*rubble pile*) o a un aggregato di ciottoli primordiali. L'esperimento CONSERT ha inoltre confermato un'elevata omogeneità strutturale su scale di 10-100 metri [26]. Dal punto di vista termico, questa elevata porosità funge da isolante eccezionale: essa impedisce al calore solare di penetrare in profondità, permettendo la conservazione di ghiacci volatili a pochi metri sotto la superficie anche dopo numerosi passaggi nel sistema solare interno [27].

L'evoluzione della superficie è strettamente legata all'attività termica e alle proprietà fotometriche del nucleo. I nuclei cometari sono tra gli oggetti più scuri del sistema solare, caratterizzati da un'albedo geometrica bassissima, generalmente compresa tra il 2% e il 6% [28]. Questo colore nerastro deriva dall'accumulo superficiale di materiale organico e composti refrattari carboniosi che rimangono come residuo dopo la sublimazione dei ghiacci. Il calore solare modifica continuamente questa morfologia: i grani di polvere espulsi più pesanti ricadono sulla superficie formando un "mantello di scorie" (*lag deposit*) inerte che, se sufficientemente ispessito, può isolare i ghiacci

sottostanti rendendo l'attività cometaria fortemente irregolare. Il paesaggio risultante è complesso e caratterizzato da pianure, campi di massi, scogliere e pozzi circolari. Questi ultimi sono stati identificati come siti di collasso di cavità sotterranee e agiscono spesso come sorgenti attive per getti di gas e polvere [27].

A livello microscopico, la composizione e lo stato fisico del ghiaccio giocano un ruolo cruciale. Il ghiaccio primordiale è solitamente in stato amorfo, una configurazione metastabile e disordinata. Quando il nucleo si riscalda raggiungendo temperature tra i 120 K e i 140 K, il ghiaccio subisce una transizione verso una struttura cristallina; tale processo rilascia energia e gas intrappolati in modo improvviso, causando i picchi di attività noti come *outbursts* [27], [29]. In generale, la chimica dei volatili vede l'acqua (H_2O) come componente dominante (circa l'80% della massa), seguita da CO e CO_2. Tuttavia, i dati di Rosetta hanno rivelato la presenza inaspettata di O_2 e N_2, traccianti che suggeriscono una formazione in ambienti estremamente freddi (temperature inferiori ai 30 K), oltre a identificare molecole organiche complesse e l'amminoacido glicina [30]. La componente solida della polvere, formata da silicati e solfuri, testimonia infine un antico rimescolamento nel disco protoplanetario tra materiali nati vicino al Sole e materiali ghiacciati delle zone esterne [30].

Specie Chimica	**Abbondanza (X/H_2O)**	**Rilevanza**
H_2O	100%	Volatile dominante e motore dell'attività [30].
CO_2	$\sim$ 4,5%	Valore di bulk stimato per il nucleo [30].
O_2	1,8 $\pm$ 0,4%	Ricalcolato su un set di dati esteso per l'intera missione [30].
CO	$\sim$ 1,7%	Media per 67P, tipica delle comete gioviane (JFCs) [30].
H_2S	$\sim$ 1,1%	La specie di zolfo più abbondante.
NH_3	0,8%	Principale portatore di azoto inorganico [30].
CH_4	0,34%	Rapporto misurato dallo strumento ROSINA [30].
N_2	0,089%	Calcolato dal valore di $(8{,}9 \pm 2{,}4) \times 10^{-4}$ [30].
Glicina	Tracce	Rilevata come sorgente distribuita dai grani di polvere [30].

Tabella 4.1. Composizione dei principali volatili di 67P.

La Chioma e le Code: L'Atmosfera Transitoria e l'Interazione Spaziale Quando il nucleo si avvicina al Sole, il riscaldamento superficiale innesca la sublimazione dei ghiacci. Questo processo non espelle solo gas, ma trascina con sé grandi quantità di particelle solide, creando attorno al nucleo un'estesa atmosfera non legata gravitazionalmente.

La chioma è l'involucro sferoidale di gas neutri e polveri che avvolge e occulta completamente il minuscolo nucleo solido. L'espansione dei gas sublimati avviene a velocità prossime a 1 km/s. A causa della bassissima gravità del nucleo, questo materiale sfugge facilmente, permettendo alla chioma di espandersi per decine di migliaia o milioni di chilometri. Dal punto di vista osservativo, la chioma domina totalmente la luminosità della cometa a causa dello scattering della luce solare, creando un forte gradiente di luminosità che rende estremamente complessa l'identificazione esatta del centroide.

Man mano che la chioma si espande, la pressione di radiazione solare e l'interazione con il vento solare "soffiano" via questo materiale, separandolo in due distinte strutture di coda che puntano sempre in direzione opposta al Sole [22]:

- **La Coda di Ioni e Plasma**: I gas della chioma vengono rapidamente ionizzati dai raggi Ultravioletto (UV) solari. Questa coda, costituita da plasma, emette luce azzurra per fluorescenza ed è governata dal vento solare. Le linee del campo magnetico interplanetario si "impigliano" nella chioma ionizzata (*magnetic field draping*), accelerando violentemente gli ioni e creando una coda rettilinea, sottile e altamente strutturata che punta nella direzione anti-solare [25].
- **La Coda di Polvere**: Costituita da particelle solide espulse dalla chioma, è modellata primariamente dalla pressione di radiazione solare. Poiché queste particelle possiedono massa, continuano a seguire orbite kepleriane individuali ma leggermente perturbate dalla pressione di radiazione stessa [31], conferendo alla coda la tipica forma curva e diffusa. Appare di colore giallastro-biancastro poiché brilla di luce solare riflessa. I grani più pesanti, che non raggiungono la velocità di fuga a causa della gravità nucleare, ricadono sulla superficie formando un mantello di scorie (dust mantle) che può portare la cometa a uno stato "dormiente" o estinto [32].

4.2 Il Problema Astrometrico: Identificazione dell'Optocentro

Come appena descritto, la composizione delle comete è molto complessa e queste possono quindi essere descritte come oggetti estesi, estremamente dinamici e intrinsecamente asimmetrici. Questo causa diverse complicazioni quando si cerca di identificare la posizione del nucleo in un'immagine digitale, costituendo un problema astrofisico e matematico di profonda complessità, che affonda le sue radici nella natura stessa dell'attività cometaria descritta in precedenza e nella fisica della dispersione della luce. Trattare una cometa alla stregua di una semplice stella puntiforme per gli algoritmi di centraggio porta inevitabilmente a misurazioni fallaci. Infatti, attraverso l'ottica di un telescopio, le stelle si presentano come sorgenti puntiformi la cui intensità spaziale è descritta unicamente dalla Point Spread Function (PSF) strumentale e atmosferica. Per le comete, intervengono fattori ben più complessi che verranno approfonditi di seguito.

4.2.1 Il Conflitto tra Optocentro e Baricentro Gravitazionale

A distanze elio-centriche e geo-centriche tipiche delle osservazioni astronomiche, il minuscolo nucleo fisico solido sottende un angolo nel cielo che è ordini di grandezza inferiore al potere risolutivo spaziale di qualsiasi telescopio ottico terrestre limitato dal seeing atmosferico. Di conseguenza, il nucleo solido non è quasi mai risolto spazialmente. Ciò che i sensori registrano è l'optocentro (o fotocentro), ovvero il baricentro dell'illuminazione della chioma interna densa di polveri e gas. La distinzione fondamentale tra optocentro e baricentro gravitazionale (il vero centro di massa, necessario per i calcoli orbitali e l'allineamento geometrico) rappresenta la principale fonte di errore sistematico. Essendo l'attività di degassamento (outgassing) guidata dall'insolazione termica, essa non è quasi mai isotropa. Il deflusso di materiale è significativamente più intenso nell'emisfero diurno, sbilanciando la chioma interna verso la direzione del Sole. Questo sposta l'optocentro rispetto alla reale posizione del nucleo solido, rendendolo un costrutto ottico dipendente dall'attività istantanea e non un punto fisico immutabile [20].

4.2.2 Campionamento Spaziale, Aliasing e Risoluzione Sub-pixel

Un altro punto che complica questo problema è il fatto che nell'ambito dell'elaborazione di immagini per la scienza cometaria, non è affatto sufficiente determinare che l'optocentro della cometa si trovi al pixel di coordinate intere. Al fine di preservare le frequenze spaziali critiche per la

morfologia, è imperativo conoscere la posizione con precisione frazionaria. L'uso sistematico di coordinate subpixel per la fase di allineamento e il successivo *stacking* è di vitale importanza ed è strettamente ancorato ai fondamenti della teoria dei segnali, specificamente al Teorema del Campionamento di Nyquist-Shannon.

In un sistema di imaging ottico ideale, il limite fisico teorico della risoluzione angolare è determinato dall'apertura, o diametro dell'obiettivo, e dalla lunghezza d'onda (Criterio di Rayleigh). Nella pratica questo limite è ridotto dalle aberrazioni introdotte dalla turbolenza atmosferica (seeing). Tuttavia, affinché il sistema di imaging (compatibilmente con le condizioni di seeing) possa sfruttare al massimo il proprio potere risolutivo, e la ricca informazione ottica analogica possa essere fedelmente preservata nel dominio digitale, il sensore di acquisizione (CCD o CMOS) deve campionare l'immagine con una dimensione dei singoli pixel adeguata alla lunghezza focale dell'ottica.

Nel progettare i sistemi ottici di imaging si è spesso costretti a scegliere tra campionare correttamente un campo visivo piccolo o sottocampionare un campo più ampio. Questo è stato un problema perfino per il Telescopio Spaziale Hubble (HST), le cui ottiche corrette possono fornire una risoluzione elevata, ma i pixel più piccoli della Planetary Camera potevano sfruttare appieno il potere risolutivo del telescopio solo su un campo visivo limitato, mentre il sensore della prima Wide Field Camera, che aveva pixel più grandi, "soffriva" di sottocampionamento.

Frequentemente nell'astronomia amatoriale, ma talvolta anche in quella professionale, si utilizzano ottiche di corta focale e sensori con pixel di grandi dimensioni per aumentare la sensibilità e massimizzare il campo visivo Field of View (FOV) allo scopo di catturare la debole ed estesa chioma cometaria e le porzioni iniziali delle code o le code intere. Questo compromesso strumentale risulta quasi sempre in immagini sottocampionate, nelle quali ogni singolo pixel copre un angolo di cielo superiore alla metà della PSF ottica.

Il Teorema di Nyquist-Shannon stabilisce un limite matematico rigido: la frequenza di campionamento spaziale deve essere rigorosamente almeno il doppio della massima frequenza spaziale (il dettaglio più piccolo) presente nel segnale ottico incidente.

In regime di sottocampionamento, il segnale (la luce della cometa) contiene frequenze spaziali più elevate di quelle che il sensore può gestire. Le alte frequenze spaziali (dettagli del nucleo, micro-bordi dei getti e filamenti) collassano su frequenze più basse, originando il fenomeno noto come *aliasing*. Questo altera irrimediabilmente la morfologia del target, introducendo artefatti quali:

- Schemi geometrici spigolosi ("stair-stepping");
- Rumore spaziale non casuale;
- Perdita netta di informazione strutturale continua o falsi dettagli.

L'impiego di coordinate subpixel è una soluzione matematica che permette di superare i limiti di risoluzione del sensore (sottocampionamento) e recuperare i dettagli reali dell'immagine (Space-Bandwidth Product (SBP)). Infatti, durante una sessione fotografica di decine di minuti, il movimento naturale della cometa o i piccoli spostamenti intenzionali del telescopio (tecnica nota come "dithering") fanno sì che la luce del nucleo colpisca punti leggermente diversi all'interno della griglia dei pixel in ogni singolo scatto.

Determinando la posizione del centroide con un'accuratezza pari o superiore a ± 0.1, gli algoritmi mappano con precisione il punto in cui il segnale ottico analogico viene proiettato sulla griglia digitale discreta del sensore. Nella successiva fase di *stacking*, algoritmi di ricampionamento (*resampling*) basati su interpolazioni di ordine superiore(come spline bicubiche, interpolazioni di Lanczos o il *Variable-Pixel Linear Reconstruction*) utilizzano questi *offset* per spalmare l'intensità su una nuova griglia virtuale ad alta risoluzione, volutamente sovracampionata (*oversampled*).

Se l'allineamento fosse rozzamente arrotondato al pixel intero (*integer shift*), le informazioni di fase ad alta frequenza verrebbero scartate. Sommare immagini con errori intra-pixel equivale ad applicare un filtro passa-basso analogico profondo, creando *blurring* spaziale non quantificabile. Per lo studio della chioma interna, dove i getti e le asimmetrie termiche coprono spesso solo

pochi pixel, un errore subpixel diluisce in modo critico il micro-contrasto nel fondo scala luminoso della chioma. Questo degrado rende vani i successivi filtri di evidenziazione morfologica, poiché l'informazione strutturale fisica è stata distrutta ancor prima di essere elaborata.

4.2.3 Asimmetrie Indotte dalla Dinamica delle Polveri e dallo Scattering

La dinamica della polvere post-emissione distorce ulteriormente la distribuzione spaziale della luce registrata nei fotogrammi FITS:

- **Pressione di Radiazione e Morfologia**: La pressione di radiazione solare modella la distribuzione spaziale della polvere spingendo le particelle in direzione antisolare. L'efficacia di tale processo dipende dalla dimensione dei grani; in 3I/ATLAS, la predominanza di particelle grandi (da decine a centinaia di micron) mitiga gli effetti della pressione radiativa, risultando in una coda corta e diffusa. A seconda della geometria di osservazione e dell'angolo di proiezione, si generano strutture a ventaglio (*fan-like structures*) o pennacchi (*plumes*) che originano dal lato diurno del nucleo [33].
- **Scattering e Fenomeni Transitori**: La dispersione della luce (scattering) segue funzioni di fase dipendenti dall'angolo di illuminazione e dalle proprietà fisiche dei grani [34]. Effetti geometrici come il *backward scattering* (all'opposizione) o il *forward scattering* (tra Terra e Sole) possono amplificare drasticamente la luminosità di porzioni specifiche della chioma [34]. L'analisi spaziale su comete come la 243P/NEAT dimostra che outburst improvvisi o asimmetrie nella chioma possono dominare il flusso luminoso locale, inducendo spostamenti repentini e non lineari dell'optocentro durante le sequenze osservative.

4.2.4 Rapporto Segnale-Rumore (SNR) e Contaminazione da Sorgenti di Fondo

L'identificazione del centro è infine ostacolata dai limiti del sensore e dalla cinematica dell'oggetto:

- **Gradienti e SNR:** L'identificazione del centro è ulteriormente ostacolata dal severo gradiente di intensità intrinseco alle chiome cometarie. Sebbene il picco centrale possa avere un elevato Rapporto Segnale-Rumore (SNR) in comete eccezionalmente brillanti, la pendenza del gradiente (che decresce approssimativamente come $1/\rho$) fa sì che i pixel adiacenti al centro siano anch'essi estremamente luminosi. Questo porta frequentemente a due scenari opposti ma ugualmente distruttivi: in comete brillanti, la saturazione del sensore (*blooming*) distrugge l'informazione del picco centrale; al contrario, in comete deboli, il centro sfuma dolcemente nel rumore di fondo (rumore di Poisson dei fotoni, rumore di lettura e corrente di buio termica) rendendo il picco statisticamente indistinguibile dai pixel adiacenti.
- **Stelle di Campo:** Le comete possiedono un moto proprio significativo rispetto al fondo cielo. Durante le sequenze prolungate, può qquindi capitare che l'optocentro si allinei invariabilmente con stelle o galassie di fondo. Gli algoritmi di centraggio, se non dotati di robuste routine per l'esclusione dei valori anomali (*outlier rejection*), tenderanno a calcolare un falso centroide basato sulla somma del flusso della cometa e della stella sovrapposta, introducendo un errore grave che corromperà l'intero risultato dell'allineamento.

4.3 Sviluppo e Valutazione di Algoritmi per la Ricerca del Nucleo

Alla luce delle intrinseche complessità morfologiche e fotometriche descritte nelle sezioni precedenti, è evidente come l'identificazione robusta dell'optocentro cometario richieda un approccio metodologico rigoroso e flessibile. Per far fronte a queste sfide, durante lo sviluppo di Kometra

[3] è stata implementata un'ampia suite di algoritmi che spazia dai classici metodi di computer vision ai più avanzati modelli di fitting astrofisico.

Al fine di valutare l'efficacia di ciascuna tecnica in modo oggettivo e iterativo, la fase di ricerca e sviluppo è stata inizialmente condotta attraverso la realizzazione di prototipi in linguaggio Python. Questa scelta architetturale ha permesso di disaccoppiare la complessa logica matematica dallo sviluppo dell'interfaccia grafica, garantendo un'iterazione veloce (*rapid prototyping*) e consentendo di testare, comparare e calibrare direttamente la bontà dei diversi algoritmi su dataset di immagini astronomiche reali.

Gli algoritmi analizzati possono essere suddivisi in due macro-categorie fondamentali, distinte in base al modo in cui interpretano e processano il segnale luminoso:

- **Approcci Empirici (Metodi diretti senza modello):** questa famiglia comprende algoritmi statistici e topologici che analizzano direttamente la matrice dei pixel grezzi. Essi calcolano il centroide tramite operazioni analitiche dirette, senza imporre alcuna assunzione preliminare sulla forma geometrica dell'oggetto luminoso. Quindi, risultano essere algoritmi deterministici e computazionalmente quasi istantanei.
- **Approcci Parametrici (Fitting tramite modello matematico):** rappresentano l'elemento centrale dell'elaborazione. Invece di limitarsi a una misurazione empirica della distribuzione luminosa, questi algoritmi adattano una funzione matematica ideale ai valori discreti dei pixel, procedendo a una minimizzazione iterativa dei residui.

4.3.1 Algoritmi Empirici

A differenza delle tecniche di ottimizzazione iterativa, questi metodi non cercano di adattare l'immagine all'interno di un'equazione matematica teorica, ma elaborano direttamente i valori grezzi dei pixel per estrarre le coordinate del baricentro. Si basano su principi statistici, geometrici o topologici, calcolando il risultato attraverso operazioni matematiche deterministiche che si risolvono in un unico passaggio computazionale.

Questa caratteristica li rende estremamente rapidi e intrinsecamente immuni ai problemi di mancata convergenza che possono affliggere i modelli più complessi. Non avendo una base matematica solida, possono essere utilizzati per fornire una stima iniziale immediata e robusta, analizzando la Region of Interest (ROI).

1. **Interpolazione Parabolica Locale:**
   L'approccio in assoluto più banale per trovare il centro consisterebbe nella semplice ricerca del pixel più luminoso dell'intera immagine. Tuttavia, restituendo coordinate puramente intere, questo metodo di base risulta di fatto inutile per le stringenti necessità di precisione dell'allineamento astronomico. Pertanto, l'algoritmo qui implementato rappresenta un'evoluzione leggermente più sofisticata, ma capace di mantenere la stessa istantaneità computazionale.

   Anziché fermarsi al valore massimo, il metodo individua prima il singolo pixel più luminoso in assoluto (il picco) e poi si concentra esclusivamente sui suoi vicini diretti per estrarre una coordinata sub-pixel. L'assunto è che la cima del profilo di luce segua una perfetta curva parabolica.

   Dal punto di vista matematico, l'algoritmo calcola la curvatura locale sull'asse orizzontale confrontando l'intensità del pixel centrale c con quello di sinistra l e di destra r, ottenendo il parametro:

   $$dxx = r + l - 2c$$

   La coordinata esatta a livello sub-pixel si ricava applicando un offset geometrico al pixel di partenza x_0:

   $$x_{sub} = x_0 - \frac{r - l}{2 \cdot dxx}$$

Lo stesso procedimento viene replicato sull'asse verticale. Questo metodo è istantaneo e ha il grande vantaggio di ignorare completamente la presenza della coda cometaria (non leggendone i pixel). Presenta però due limiti strutturali: la vulnerabilità alla saturazione (se il centro è "bruciato", presentando più pixel adiacenti con il medesimo valore massimo, la curvatura si azzera bloccando la formula) e la sensibilità ai falsi segnali (un singolo raggio cosmico o un pixel difettoso del sensore viene interpretato erroneamente come il vero nucleo).

È importante precisare che, sebbene sfrutti l'equazione di una parabola, questo algoritmo ricade a pieno titolo tra gli approcci empirici e non tra quelli parametrici. Il suo impiego, infatti, è puramente topologico e locale: il metodo non tenta di approssimare l'intera morfologia della cometa (come farebbe un modello globale con il gas della chioma), ma utilizza l'interpolazione parabolica unicamente per stimare la posizione del vertice geometrico confinato tra tre pixel discreti adiacenti. Essendo basato su operazioni matematiche dirette, chiuse e non iterative sui pixel grezzi, esso soddisfa rigorosamente la definizione di metodo empirico.

2. **Baricentro Luminoso Globale:**
   Si tratta del metodo astrometrico più classico. Interpreta l'immagine valutando l'intensità luminosa di ogni pixel come se fosse una massa fisica, con lo scopo di trovare il punto di equilibrio geometrico dell'intera area analizzata. Le coordinate esatte del centro vengono ricavate calcolando una media ponderata: si divide la distribuzione spaziale della luce lungo i due assi (i momenti di primo ordine m_{10} e m_{01}) per la somma totale dell'intensità luminosa (il momento di ordine zero m_{00}). La formula applicata è:

   $$x_c = \frac{m_{10}}{m_{00}}, \quad y_c = \frac{m_{01}}{m_{00}}$$

   Sebbene sia rapido e stabile su sorgenti puntiformi e simmetriche come le stelle, si rivela spesso inadeguato per le comete. Poiché il calcolo prende in considerazione l'intera area senza applicare filtri, la vasta quantità di pixel deboli che compongono la coda asimmetrica sbilancia l'equazione, trascinando il baricentro lontano dal nucleo reale.

3. **Baricentro a Soglia Statistica Adattiva:**
   Concepito come un'evoluzione diretta del baricentro luminoso, questo algoritmo nasce per limitare l'influenza asimmetrica della coda e del rumore di fondo. L'algoritmo applica un filtro preventivo isolando esclusivamente i pixel più luminosi. La soglia di taglio viene determinata dinamicamente su base statistica calcolando la media μ e la deviazione standard σ dell'area analizzata. Tutti i pixel con un'intensità inferiore o uguale al valore di soglia:

   $$\text{Threshold} = \mu + 3\sigma$$

   vengono forzati a zero. Sull'immagine così mascherata, il sistema procede poi con il calcolo dei momenti spaziali tramite le formule del baricentro globale.

   Questo metodo offre una stabilità nettamente superiore per i bersagli cometari. Eliminando matematicamente i pixel a bassa intensità che compongono la maggior parte della coda e del fondo cielo, impedisce che queste aree estese alterino in modo significativo la posizione del baricentro. Pur mantenendo un'elevata rapidità computazionale, la sua efficacia resta dipendente dal contrasto generale dell'immagine e dall'assenza di altri elementi luminosi che potrebbero superare accidentalmente la soglia statistica.

4. **Intersezione dei Gradienti Radiali:**
   Questo metodo cambia radicalmente prospettiva: non valuta la quantità assoluta di luce, ma si concentra sulla sua variazione spaziale, ovvero sulle pendenze del profilo luminoso. Assumendo che un corpo celeste presenti una simmetria radiale, tracciando delle rette lungo la direzione di massima pendenza in ogni punto, queste convergeranno tutte verso il centro esatto. L'algoritmo utilizza i filtri di Sobel per estrarre i gradienti g_x e g_y. Per dare maggiore importanza ai contorni netti rispetto al rumore, a ogni pixel viene assegnato un peso proporzionale alla magnitudine del gradiente:

   $$w = \sqrt{g_x^2 + g_y^2}$$

Il problema geometrico di far intersecare tutte queste rette si traduce nella risoluzione di un sistema di equazioni lineari sovradeterminato, calcolato tramite il metodo dei minimi quadrati. L'algoritmo costruisce e risolve un sistema matriciale nella forma:

$$A\mathbf{x} = B$$

dove la matrice A e il vettore B contengono le somme ponderate delle coordinate spaziali moltiplicate per i rispettivi gradienti direzionali. La risoluzione del sistema restituisce il vettore incognito $\mathbf{x}$, il quale definisce le coordinate finali del centroide (x_c, y_c).

Tale approccio risulta altamente efficace nella gestione dei pixel centrali saturati: le aree di saturazione, essendo piatte e a gradiente nullo, vengono ignorate a vantaggio dei profili di emissione adiacenti. L'algoritmo presenta però una vulnerabilità alle marcate asimmetrie della chioma interna, che rischiano di far divergere le rette di calcolo.

5. **Mappatura di Simmetria a Finestra Mobile:**
A differenza dei metodi precedenti, non cerca semplicemente il punto più luminoso o il bilanciamento delle masse, ma esplora l'immagine alla ricerca del punto geometricamente più rotondo e bilanciato. L'algoritmo fa scorrere una piccola finestra quadrata, definita setaccio, su tutta la regione di interesse. Per ogni singola posizione analizzata, il sistema estrae il blocco di pixel, lo ruota di 180 gradi e calcola la differenza assoluta tra il blocco originale e la sua versione capovolta. Dal punto di vista matematico, la magnitudine totale della luce presente nel blocco viene confrontata con la differenza generata dalla rotazione per ricavare un indice di simmetria locale. La formula penalizza fortemente le zone che cambiano aspetto se capovolte:

$$\text{symmetry} = \max\left(0, 1 - \frac{\sum |S - S_{rot}|}{\text{magnitude}}\right)$$

Successivamente, l'algoritmo assegna un peso specifico a quella coordinata spaziale moltiplicando la quantità di luce originaria per il quadrato della simmetria trovata:

$$W = \text{magnitude} \cdot \text{symmetry}^2$$

Le coordinate finali vengono ricavate calcolando un classico baricentro spaziale, utilizzando però questi nuovi pesi strutturali W. Questo approccio è estremamente sofisticato: quando la finestra passa sulla coda (fortemente asimmetrica), l'errore di rotazione è altissimo e il peso assegnato crolla a zero, permettendo all'algoritmo di concentrarsi solo sulla chioma interna. Il compromesso primario per ottenere questa resilienza morfologica è l'onere computazionale: a causa dei doppi cicli di scansione annidati e delle continue rotazioni matriciali, questo metodo risulta l'algoritmo empirico più lento tra quelli presentati. Un'implementazione prototipale in linguaggio Python di questa logica di setacciamento è illustrata in Figura 4.2.

4.3.2 Algoritmi Parametrici

A differenza dei metodi empirici, le tecniche parametriche non si limitano a estrarre una misura puntuale dai pixel grezzi, ma mirano a comprendere e ricostruire l'intera distribuzione spaziale della luce. Questi algoritmi forzano un'equazione matematica ideale (una superficie tridimensionale definita da parametri incogniti) ad adattarsi all'immagine reale, minimizzando progressivamente i residui attraverso un processo iterativo.

Per far fronte all'eterogeneità morfologica dei corpi celesti e risolvere le criticità specifiche poste dalle comete, l'architettura software implementa una progressione di modelli via via più sofisticati. Di seguito sono presentati gli algoritmi sviluppati, ordinati dai modelli geometrici di base fino alle più complesse superfici asimmetriche:

1. **Adattamento Gaussiano Bidimensionale:**
Il primo algoritmo parametrico presentato è l'adattamento gaussiano bidimensionale. Questo approccio crea un modello matematico tridimensionale a forma di campana e cerca di

```python
def find_center_mappatura_simmetria(roi, x_off, y_off, dimensione_setaccio=5):
    h, w = roi.shape
    offset = dimensione_setaccio // 2

    weighted_x = 0.0
    weighted_y = 0.0
    total_w = 0.0

    # Analisi locale di simmetria rotazionale
    for y in range(offset, h - offset):
        for x in range(offset, w - offset):
            setaccio = roi[y-offset:y+offset+1, x-offset:x+offset+1]
            magnitudine = np.sum(setaccio)
            if magnitudine <= 1e-9: continue

            # Confronto con la versione ruotata di 180 gradi
            setaccio_ruotato = np.rot90(setaccio, 2)
            differenza = np.sum(np.abs(setaccio - setaccio_ruotato))

            simmetria = max(0, 1.0 - (differenza / magnitudine))
            peso = magnitudine * (simmetria**2)

            weighted_x += (x + 0.5) * peso
            weighted_y += (y + 0.5) * peso
            total_w += peso

    if total_w <= 1e-9:
        return float('inf'), float('inf'), float('inf'), 0.0

    xc, yc = weighted_x / total_w, weighted_y / total_w
    return x_off + xc, y_off + yc, 0.0, calcola_r2_classico(roi, xc, yc)
```

Figura 4.2. Prototipo Python dell'algoritmo di mappatura di simmetria, illustrante la logica dei cicli annidati e della rotazione matriciale per l'estrazione della simmetria locale.

adattarlo alla distribuzione di luce reale dei pixel attraverso un processo iterativo di minimizzazione dei residui. A differenza di una campana perfettamente sferica, l'algoritmo permette a questa superficie di assumere una forma ellittica e di ruotare su se stessa per seguire l'inclinazione dell'oggetto inquadrato.

Dal punto di vista matematico, il motore di ottimizzazione fa variare progressivamente parametri fondamentali come le coordinate del centro (x_c, y_c), le deviazioni standard lungo i due assi σ_x e σ_y, e l'angolo di rotazione θ. Il modello valutato per ogni pixel calcola i coefficienti direzionali a, b e c, assumendo la forma base:

$$f(x, y) = A \cdot \exp\left(-(ax^2 + by^2 + 2cxy)\right) + \text{offset}$$

La minimizzazione della differenza tra questo modello ideale e l'immagine reale avviene tramite la funzione dei minimi quadrati di SciPy, utilizzando una metrica di perdita denominata *soft_l1*. Questa scelta specifica serve a garantire robustezza, riducendo pesantemente l'influenza matematica di pixel anomali estremamente devianti, come quelli generati dall'impatto di raggi cosmici sul sensore.

Nel suo comportamento reale, questo metodo rappresenta lo standard per l'astrometria stellare, garantendo misurazioni di altissima precisione su sorgenti puntiformi. Tuttavia, applicato alle comete, mostra dei limiti fisici intrinseci: la funzione esponenziale della gaussiana decade verso lo zero in modo estremamente rapido allontanandosi dal picco centrale. Di conseguenza, il modello non riesce a descrivere correttamente il decadimento lento e diffuso tipico del gas della chioma, rischiando di ignorare informazioni fotometriche periferiche essenziali.

2. **Scomposizione Nucleo-Chioma:**
Questo algoritmo affronta la centratura da una prospettiva prettamente fisica, cercando di scomporre l'immagine nelle due componenti luminose fondamentali di una cometa: il nucleo solido, inevitabilmente sfocato dall'ottica del telescopio, e l'atmosfera di gas diffuso circostante. L'obiettivo è isolare matematicamente la posizione esatta del falso nucleo separandolo dall'alone della chioma.

   Dal punto di vista matematico, il sistema fonde due equazioni distinte basate sulla distanza radiale dal centro stimato, definita come $\rho = \sqrt{(x - x_c)^2 + (y - y_c)^2}$. Il modello sovrappone una classica Gaussiana compatta, che modella il picco del nucleo, a una curva a decadimento radiale lento per simulare la chioma:

$$f(x, y) = \text{bg} + A_{\text{nuc}} \cdot \exp\left(-\frac{\rho^2}{2\sigma^2}\right) + A_{\text{coma}} \cdot \frac{1}{\sqrt{\rho^2 + \sigma^2}}$$

   Un ottimizzatore iterativo basato sui minimi quadrati interviene per far variare progressivamente le coordinate incognite (x_c, y_c), il livello di fondo cielo bg e le rispettive ampiezze luminose A_{nuc} e A_{coma}. Il processo si arresta quando la somma di queste due forme geometriche ricalca con la massima fedeltà possibile la distribuzione di luce reale dei pixel.

   Nel suo comportamento reale, questo metodo si distingue per l'elevato rigore analitico ed è particolarmente prezioso quando si necessita di misurare non solo la posizione, ma anche quanta luce provenga strettamente dal nucleo rispetto al gas. Essendo basato su un motore di fittaggio iterativo, risulta intrinsecamente più lento e oneroso per il processore rispetto alle semplici equazioni algebriche. Inoltre, costruendo la componente della chioma sulla pura distanza radiale ρ, l'algoritmo impone una simmetria sferica di base all'alone gassoso; di conseguenza, pur separando perfettamente il nucleo dalla chioma interna, non riesce a modellare in modo dinamico la forte direzionalità di un'eventuale coda di polveri molto sviluppata.

3. **Profilo a Legge di Potenza Monodimensionale:**
Per superare i limiti strutturali dell'esponenziale gaussiano, questo approccio introduce l'uso della legge di potenza. Semplifica il problema spaziale estraendo due singole "fette" monodimensionali dall'immagine, ovvero la singola riga orizzontale e la singola colonna verticale che si incrociano in corrispondenza del pixel più luminoso. L'algoritmo analizza questi due profili in modo completamente separato.

Dal punto di vista matematico, su queste linee viene applicato un modello fotometrico basato su una legge di potenza asimmetrica, derivata dalla letteratura scientifica specifica per l'analisi fotometrica della chioma cometaria [35]. La funzione fitta i dati calcolando il flusso luminoso teorico:

$$f(x) = \frac{A}{|x - \delta|^n + a} + \text{bg}$$

dove δ rappresenta la coordinata del centro sub-pixel incognito, A l'ampiezza del segnale, a una costante di addolcimento per evitare divisioni per zero e bg il livello del fondo cielo. La caratteristica cruciale è la sua asimmetria strutturale: l'esponente di decadimento n non è unico, ma viene diviso in n_{neg} e n_{pos}, a seconda che il punto valutato si trovi prima o dopo il centro stimato. Inoltre, l'algoritmo sovracampiona matematicamente ogni singolo pixel suddividendolo in dieci sotto-intervalli e mediandone i risultati.

Nel suo comportamento reale, questo metodo si rivela fisicamente molto più accurato della gaussiana nel catturare la differenza netta di decadimento tra la parte frontale della chioma e la coda. Tuttavia, il suo limite principale risiede nella scomposizione monodimensionale: elaborando gli assi separatamente, perde la percezione globale della struttura 2D. Questo lo rende meno stabile se la morfologia della cometa si sviluppa lungo assi diagonali.

4. **Profilo a Legge di Potenza Radiale Troncata:**
   Questo algoritmo estende il concetto della legge di potenza all'intero piano bidimensionale. In questo modo, analizza l'intera griglia di pixel simultaneamente, cercando l'adattamento ottimale su tutta la regione di interesse.

   Dal punto di vista matematico, il modello assume una simmetria radiale di base attorno al fotocentro. Per ciascuna coordinata spaziale, viene calcolata la distanza euclidea r dal centro stimato (x_c, y_c), modellando l'intensità luminosa secondo l'equazione:

   $$f(x, y) = A \cdot (\max(r, 0.5))^{-n} + \text{bg}$$

   Un elemento distintivo di questo approccio è l'impiego della funzione di *massimo* tra la distanza reale r e un valore soglia pari a 0.5 pixel. Tale artificio analitico consente di troncare il calcolo in prossimità del nucleo, prevenendo la divisione per zero e la conseguente divergenza della funzione verso l'infinito.

   Operativamente, questo metodo ricostruisce l'intera morfologia dell'oggetto, garantendo una stabilità algoritmica complessiva superiore rispetto alla scomposizione monodimensionale. Tuttavia, imponendo all'esponente n un valore univoco e isotropo in tutte le direzioni radiali, il modello sacrifica la flessibilità necessaria per adattarsi a marcate asimmetrie direzionali.

5. **Profilo Radiale con Addolcimento Centrale:**
   Questo algoritmo, rispetto a quelli precedenti, cerca di gestire in modo analiticamente più rigoroso il problema della singolarità centrale. Anziché imporre un troncamento artificiale alla distanza, il metodo introduce un parametro di regolarizzazione (o *softening*) strutturale direttamente all'interno dell'equazione.

   Quindi, il modello calcola il quadrato della distanza radiale, $r^2 = (x - x_c)^2 + (y - y_c)^2$, e vi somma il quadrato di un parametro di softening r_0:

   $$f(x, y) = A \cdot (r^2 + r_0^2)^{-n/2} + \text{bg}$$

   Questa formulazione genera un profilo ampiamente diffuso in astrofisica: in corrispondenza del centro esatto, la funzione non diverge, bensì descrive un nucleo appiattito e regolare, per poi raccordarsi asintoticamente a un decadimento a legge di potenza caratterizzato da ali diffuse.

   Nel suo comportamento reale, questo metodo risulta estremamente stabile e fisicamente rigoroso, garantendo una transizione fluida tra il nucleo compatto e il gas circostante senza salti matematici. Tuttavia, imponendo una simmetria radiale perfetta, presume che la distribuzione di luce sia perfettamente circolare, faticando ad assecondare l'ellitticità dell'oggetto. Un'implementazione prototipale in Python di questo modello base a simmetria radiale è mostrata in Figura 4.3.

```
    def fit_profilo_radiale_addolcito(roi, x_off, y_off):
        h, w = roi.shape
        x_grid, y_grid = np.meshgrid(np.arange(w), np.arange(h))

        # Stima iniziale del background e parametri (p0)
        bg = np.percentile(roi, 5)
        # p = [xc, yc, Ampiezza, esponente (n), softening (r0), bg]
        p0 = [w / 2.0, h / 2.0, np.max(roi) - bg, 1.0, 1.5, bg]

        # Definizione del modello a legge di potenza addolcita
        def modello_radiale(p, x, y):
            r_sq = (x - p[0])**2 + (y - p[1])**2
            return p[2] * (r_sq + p[4]**2)**(-p[3] / 2.0) + p[5]

        # Funzione obiettivo per il calcolo dei residui
        def residui(p, x, y, z):
            return (modello_radiale(p, x, y) - z).ravel()

        # Ottimizzazione ai minimi quadrati (Levenberg-Marquardt o
            Trust Region)
        res = least_squares(residui, p0, args=(x_grid, y_grid, roi),
                            bounds=([0, 0, 0, 0.1, 0.1, -np.inf],
                                    [w, h, np.inf, 5.0, 20.0, np.inf]),
                            ftol=1e-5)

        return x_off + res.x[0], y_off + res.x[1]
```

Figura 4.3. Implementazione del modello a legge di potenza con simmetria radiale e parametro di addolcimento strutturale.

6. **Profilo Radiale Addolcito con Vincoli Fisici:**
   Si tratta di una rivisitazione strettamente vincolata del modello precedente. Abbandona flessibilità esterne per garantire la massima aderenza ai profili fotometrici teorici descritti nella letteratura scientifica.

   La formula matematica è la seguente:

   $$f(x,y) = A \cdot ((x-x_c)^2 + (y-y_c)^2 + r_0^2)^{-n/2} + \text{bg}$$

   La principale differenza risiede nei limiti operativi imposti all'ottimizzatore. Infatti, i confini di ricerca sono calibrati in modo stringente: l'esponente massimo viene limitato a 4 e il raggio di addolcimento a 15 pixel.

   Questa restrizione forza il sistema a rispettare rigorosamente la fisica dei gas cometari standard, impedendogli di assumere valori matematicamente perfetti ma fisicamente impossibili. Garantisce un'elevata stabilità su immagini molto rumorose, a costo però di rinunciare alla flessibilità necessaria per mappare chiome fortemente ellittiche.

7. **Profilo Ellittico Addolcito e Orientabile:**
   Questo algoritmo fonde i vantaggi del modello ad addolcimento centrale con una maggiore flessibilità geometrica, consentendo alla superficie analitica di superare il vincolo della simmetria circolare. In questo modo, il modello può orientarsi e deformarsi nello spazio bidimensionale per adattarsi a distribuzioni di luce allungate.

   Dal punto di vista matematico, si esegue prima una trasformazione del sistema di riferimento, ruotando le coordinate di un angolo θ:

   $$x_r = dx\cos(\theta) + dy\sin(\theta)$$

   $$y_r = -dx\sin(\theta) + dy\cos(\theta)$$

   Poi, su questo piano ruotato viene introdotto un fattore di scala q sull'asse delle ordinate per schiacciare il profilo:

   $$f(x,y) = A \cdot \left(x_r^2 + \left(\frac{y_r}{q}\right)^2 + r_0^2 \right)^{-n/2} + \text{bg}$$

   Nel suo comportamento reale, l'algoritmo si rivela particolarmente efficace nel tracciare comete con isofote ellittiche, allineandosi autonomamente lungo l'asse principale dell'emissione luminosa. Il suo limite intrinseco risiede tuttavia nella simmetria speculare imposta dalla formulazione: trattando l'allungamento in modo strettamente bilaterale, il modello risulta inadeguato nel riprodurre l'emissione di una coda cometaria che si sviluppi in maniera marcatamente unidirezionale.

8. **Profilo Ibrido Asimmetrico-Rotazionale:**
   Questo algoritmo si differenzia dai precedenti introducendo per la prima volta un'asimmetria strutturale bidimensionale. In particolare, fonde la capacità di orientamento spaziale del piano con la differenziazione direzionale del decadimento luminoso propria dei modelli monodimensionali.

   Il modello esegue una rotazione del sistema di coordinate di un angolo θ. La novità risiede nella gestione dell'esponente lungo l'asse ruotato x_r: seleziona dinamicamente un esponente n_m diverso a seconda che il pixel valutato si trovi "davanti" o "dietro" il fotocentro:

   $$n_m = \begin{cases} n_{pos}, & \text{se } x_r \geq 0 \\ n_{neg}, & \text{se } x_r < 0 \end{cases}$$

   L'equazione continua diventa:

   $$f(x,y) = A \cdot (x_r^2 + y_r^2 + r_0^2)^{-n_m/2} + \text{bg}$$

   Rappresenta un salto di qualità per la centratura: orientando l'asse x_r lungo la direzione della coda, descrive contemporaneamente il rapido decadimento della chioma frontale e la lunga estensione del gas opposto, senza farsi sbilanciare dalla massa luminosa. Il limite è che l'asimmetria viene concessa solo lungo il singolo asse direzionale principale.

9. **Profilo Ortogonale Anisotropico:**
   Questo approccio supera la necessità di ruotare il sistema di riferimento. Introduce, al contrario, il concetto di anisotropia spaziale, consentendo al profilo luminoso di decadere con gradienti del tutto indipendenti lungo gli assi orizzontale e verticale dell'immagine.

   Il modello valuta indipendentemente le componenti X e Y e utilizza due esponenti di decadimento separati, n_x e n_y, sommandoli al denominatore:

$$f(x,y) = \frac{A}{((x-x_c)^2 + r_0^2)^{n_x/2} + ((y-y_c)^2 + r_0^2)^{n_y/2}} + \text{bg}$$

   Questa formulazione offre un eccellente compromesso tra flessibilità e stabilità, riuscendo a modellare isofote ovalizzate e allineate agli assi del sensore senza l'onere computazionale di un angolo di rotazione aggiuntivo. Tuttavia, se la cometa presenta una forte asimmetria diagonale, l'algoritmo sarà costretto a mediare i valori degli esponenti, riducendo conseguentemente la propria efficacia.

10. **Profilo Asimmetrico a Quadranti:**
    Questa sintesi evolutiva partiziona lo spazio in quattro quadranti indipendenti centrati sul nucleo, permettendo al modello di decadere con quattro gradienti distinti. Ciò consente di gestire variazioni repentine e asimmetriche nella distribuzione spaziale della luminosità.

    L'algoritmo adotta una formulazione a cuspide basata su una pura legge di potenza, ottimizzata per tracciare con estrema precisione il picco acuto del nucleo. Il sistema seleziona dinamicamente l'esponente in base al segno della distanza: se un pixel si trova a destra del centro viene utilizzato n_{x+}, se a sinistra n_{x-}, e analogamente per n_{y+} e n_{y-}. L'intensità luminosa viene modellata sommando i contributi assoluti dei due assi:

$$f(x,y) = \frac{A}{|x-x_c|^{n_x} + |y-y_c|^{n_y} + r_0} + \text{bg}$$

    Tale struttura matematica garantisce una perfetta continuità analitica in corrispondenza del fotocentro, eliminando eventuali discontinuità tra i quadranti e generando un profilo centrale decisamente più acuto rispetto ai modelli con addolcimento quadratico. In questo contesto, il parametro di *softening* r_0 interviene come termine lineare al denominatore, assolvendo la duplice funzione di calibrare l'ampiezza del picco e prevenire la singolarità matematica.

11. **Modello Asimmetrico a Residui Pesati:**
    Questo approccio rappresenta la summa algoritmica avanzata delle tecniche finora discusse, coniugando la precisione morfologica dell'asimmetria a quattro gradienti con una componente fondamentale per la stabilità spaziale: la maschera di pesatura. L'obiettivo primario è formulare un modello fisicamente accurato e al contempo intrinsecamente immune alle perturbazioni periferiche.

    Il cuore dell'algoritmo mantiene la divisione a quattro quadranti (n_{x+}, n_{x-}, n_{y+}, n_{y-}). La vera innovazione risiede nella valutazione del fit. Il sistema genera una finestra gaussiana bidimensionale statica, centrata al centro del ritaglio $(w/2, h/2)$:

$$W(x,y) = \exp\left(-\frac{(x-w/2)^2 + (y-h/2)^2}{2\sigma_w^2}\right)$$

    Questa maschera decade dolcemente verso lo zero sui bordi. Durante l'ottimizzazione, i residui vengono moltiplicati per questa maschera e minimizzati tramite la funzione di perdita robusta *soft_l1*.

    La maschera di pesatura ha il compito di annullare l'influenza dei bordi (rendendo di fatto l'algoritmo insensibile alla presenza di stelle di campo vicine), la metrica *soft_l1* mitiga l'impatto di eventuali raggi cosmici sovrapposti al nucleo, e l'equazione a quattro gradienti modella fedelmente la transizione asimmetrica tra chioma e coda. La complessità architetturale di questa sintesi matematica, che unisce l'asimmetria spaziale alla pesatura dei residui, è illustrata nel diagramma di Figura 4.4.

```python
def fit_modello_asimmetrico_pesato(roi, x_off, y_off):
    h, w = roi.shape
    y_grid, x_grid = np.meshgrid(np.arange(h), np.arange(w), indexing='ij')

    bg = np.percentile(roi, 10)
    amp = np.max(roi) - bg

    # 1. MASCHERA DI PESATURA: finestra gaussiana per ignorare i bordi
    sigma_w = min(h, w) / 3.5
    peso = np.exp(-((x_grid - w/2)**2 + (y_grid - h/2)**2) / (2 * sigma_w**2))

    # Parametri: [xc, yc, A, nx+, nx-, ny+, ny-, r0, bg]
    p0 = [w/2, h/2, amp, 1.0, 1.0, 1.0, 1.0, 1.5, bg]

    # 2. MODELLO ASIMMETRICO: pendenze indipendenti per quadrante
    def modello_asimmetrico(p, x, y):
        dx, dy = x - p[0], y - p[1]

        nx = np.where(dx >= 0, p[3], p[4])
        ny = np.where(dy >= 0, p[5], p[6])

        termine_x = (dx**2 + p[7]**2)**(nx / 2.0)
        termine_y = (dy**2 + p[7]**2)**(ny / 2.0)

        return p[2] * (1.0 / (termine_x + termine_y)) + p[8]

    # 3. RESIDUI PESATI: ignora le anomalie periferiche
    def residui(p):
        modello = modello_asimmetrico(p, x_grid, y_grid)
        return ((modello - roi) * peso).ravel()

    # 4. OTTIMIZZAZIONE ROBUSTA: minimizzazione con metrica soft_l1
    res = least_squares(residui, p0,
                        bounds=([0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.5, -np.inf],
                                [w, h, np.inf, 5.0, 5.0, 5.0, 5.0, 10.0, np.inf]),
                        loss='soft_l1', f_scale=amp*0.1, ftol=1e-5)

    return x_off + res.x[0], y_off + res.x[1]
```

Figura 4.4. Prototipo Python del Modello Asimmetrico a Residui Pesati, caratterizzato dalla scomposizione in quattro quadranti asimmetrici e dalla funzione di perdita *soft-L1* pesata spazialmente.

4.3.3 Risultati Sperimentali e Benchmarking dei Modelli

Definita la rosa degli algoritmi per l'identificazione del centro cometario, si è reso necessario stabilire un metodo di confronto rigoroso per selezionare la strategia ottimale da implementare nell'applicativo. La valutazione oggettiva delle prestazioni in astrometria cometaria rappresenta, tuttavia, una sfida metodologica complessa, principalmente a causa dell'assenza di un *Ground Truth* assoluto nelle immagini reali.

Un primo approccio ha previsto l'impiego di metriche standard per valutare la bontà di adattamento dei modelli matematici al profilo di luce, prima fra tutte il coefficiente di determinazione (R^2). L'ipotesi di partenza postulava che un algoritmo capace di approssimare perfettamente l'immagine avrebbe restituito le coordinate più accurate. Tuttavia, questa via è stata rapidamente scartata per un vizio concettuale di fondo: la metrica R^2 premia i modelli che descrivono con precisione l'intera distribuzione luminosa, includendo la chioma estesa e la coda asimmetrica. Dal punto di vista astrometrico, l'obiettivo non è descrivere fedelmente la morfologia del gas, bensì individuarne l'origine fisica (il nucleo). Inoltre, tale metrica rendeva impossibile un confronto equo tra i metodi parametrici (valutabili tramite una funzione di fit) e i metodi euristici classici (privi di un modello analitico di riferimento).

Sono state successivamente prese in considerazione altre metriche (come la FWHM) e strategie operative basate sullo *stacking* e sull'applicazione del filtro Larson-Sekanina [7]). L'intento era verificare quale metodo restituisse un'immagine fusa con il nucleo centrato nel modo più accurato, minimizzando la comparsa di un "dipolo" centrale (sintomo visivo di un disallineamento nei frame originali). Queste tecniche, pur validando qualitativamente i risultati, si sono rivelate instabili e incapaci di fornire valori quantitativi robusti per trarre conclusioni oggettive sull'efficacia degli algoritmi.

Preso atto dell'inaffidabilità dell'autovalutazione basata sulle immagini reali, si è esplorata la tecnica del "Consenso". L'idea consisteva nel far elaborare la medesima immagine a un ampio pool di algoritmi, estraendo poi la media o la mediana delle coordinate risultanti, con la speranza che gli errori sistematici dei singoli approcci si elidessero a vicenda. Questo paradigma, tuttavia, per sua stessa costruzione, tende a convergere verso un comportamento medio piuttosto che premiare il metodo fisicamente più corretto. Di conseguenza, se la maggioranza degli algoritmi ottiene risultati insufficienti (ad esempio subendo una forte trazione sistematica verso la coda), anche la stima di consenso risulterà viziata.

Si è reso quindi necessario creare un ambiente di test totalmente controllato, in cui le coordinate esatte del nucleo fossero note a priori. Si è optato per la generazione di un dataset sintetico di immagini FITS (400×400 pixel a 16-bit), programmato per rispettare rigorosamente la fisica e la morfologia cometaria reale, pur mantenendo il centro matematico assoluto ancorato al *Ground Truth*. Il generatore implementa una simulazione basata sulla fisica che descrive la dinamica delle polveri cometarie: le particelle vengono espulse dal nucleo in direzione del Sole, ma la pressione di radiazione solare le frena e le respinge all'indietro, formando una parabola asimmetrica. Per ricreare questa morfologia, la simulazione unisce:

- Un nucleo sub-pixel ellittico e ruotato, per evitare una perfetta sfericità irrealistica.
- Un gradiente di emissione dipendente dall'angolo di fase, che sbilancia fisicamente il picco di luce verso la direzione solare senza traslare l'origine matematica della funzione.
- Un inviluppo esponenziale asimmetrico per simulare la pressione di radiazione e la conseguente formazione della coda.

A questo modello sono stati poi sovrapposti densi campi stellari gaussiani, un gradiente lineare per il fondo cielo, rumore variabile di tipo Poissoniano e Gaussiano, e artefatti strumentali (*hot* e *dead pixels*).

Per garantire una solidità statistica al test, sono state generate 60 immagini, suddivise in tre classi di difficoltà specifiche, ciascuna mirata a testare un punto di debolezza degli algoritmi:

- **Ideale (20 immagini):** Comete con un buon rapporto segnale/rumore (SNR), geometria non esasperata e poche stelle di campo. Rappresenta la linea di base valutativa.

- **Debole/Noise (20 immagini):** Comete estremamente deboli immerse in un fondo cielo inquinato, caratterizzate da un rumore di lettura elevato. Questo set verifica la robustezza degli algoritmi nell'evitare la convergenza su falsi picchi luminosi.
- **Asimmetrica (20 immagini):** Costituisce lo stress-test geometrico. Presenta nuclei fortemente allungati, un'emissione luminosa marcatamente sbilanciata in avanti (estremo effetto di fase) e code estese sovrapposte a campi stellari densi. Valuta la capacità del modello di identificare l'effettiva origine geometrica, mitigando il bias indotto dalla distribuzione asimmetrica del gas e delle polveri.

Avendo finalmente a disposizione un *Ground Truth* inequivocabile, la valutazione è stata strutturata isolando le variabili prestazionali attraverso tre metriche rigorose:

1. **Accuratezza Pura (Errore):** La distanza euclidea, espressa in pixel, tra la coordinata calcolata dall'algoritmo e il vero centro geometrico.
2. **Stabilità allo Zoom:** Per ogni immagine, l'algoritmo è stato forzato a operare su finestre di ritaglio (ROI) di dimensioni crescenti: raggi di 35, 50 e 65 pixel. La deviazione standard dei risultati indica quanto il modello si faccia ingannare dalla presenza di una maggiore porzione di coda o di fondo cielo ai bordi.
3. **Stabilità all'Offset:** Mantenendo fissa la dimensione del ritaglio, si è simulata l'imprecisione umana generando dei *click* di innesco casualmente sfalsati rispetto al vero centro (fino a 10 pixel di errore). Lo *spread* risultante misura l'immunità del modello all'operatore, ovvero la sua capacità di convergere sempre sul medesimo punto indipendentemente dalla stima iniziale.

		Test Zoom (Crop Var.)		**Test Offset (Click Var.)**			
Pos.	**Algoritmo**	**Err. (px)**	**Spread (px)**	**Err. (px)**	**Spread (px)**	**Score (px)**	**Tempo (ms)**
1	Modello Asimmetrico a Residui Pesati	2.45	±1.27	3.10	±1.66	**4.242**	348.22
2	Profilo Asimmetrico a Quadranti	4.62	±0.34	4.71	±0.39	**5.029**	212.23
3	Scomposizione Nucleo-Chioma	5.45	±0.32	5.49	±0.41	**5.831**	42.16
4	Profilo a Potenza Radiale Troncata	5.32	±0.51	5.73	±0.59	**6.076**	77.44
5	Profilo Radiale Addolcito (Vincolato)	6.25	±0.32	6.30	±0.16	**6.516**	170.34
6	Profilo Radiale con Addolcimento Centrale	6.26	±0.33	6.41	±0.20	**6.596**	215.97
7	Profilo Ortogonale Anisotropico	6.51	±0.45	6.56	±0.27	**6.891**	296.88
8	Baricentro Luminoso Globale	4.18	±0.72	6.49	±3.66	**7.522**	0.81
9	Mappatura di Simmetria a Finestra Mobile	4.33	±0.67	6.79	±3.84	**7.817**	232.35
10	Baricentro a Soglia Statistica Adattiva	8.70	±4.42	9.63	±3.13	**12.940**	0.73
11	Profilo Ellittico Addolcito e Orientabile	12.39	±6.11	11.20	±3.55	**16.623**	844.02
12	Intersezione dei Gradienti Radiali	16.47	±11.48	19.14	±7.15	**27.117**	1.17
13	Profilo Ibrido Asimmetrico-Rotazionale	15.57	±10.35	18.96	±11.73	**28.308**	386.29
14	Adattamento Gaussiano Bidimensionale	18.87	±15.12	21.56	±16.87	**36.205**	68.53
15	Interpolazione Parabolica Locale	42.94	±25.45	45.46	±11.95	**62.894**	0.59
16	Profilo a Potenza Monodimensionale	42.53	±26.29	45.33	±11.90	**63.024**	640.40

Tabella 4.2. Risultati completi del benchmark aggiornato sul dataset sintetico. Gli algoritmi sono ordinati in base allo *Score Definitivo*. I tempi di esecuzione sono riportati a scopo informativo.

I risultati dei test eseguiti sul dataset sintetico sono riassunti nella Tabella 4.2. Prima di addentrarsi nell'analisi delle prestazioni, è fondamentale chiarire il significato delle metriche riportate nelle singole colonne, al fine di interpretare correttamente il comportamento degli algoritmi. L'accuratezza assoluta, o errore medio, rappresenta la distanza euclidea tra le coordinate restituite dall'algoritmo e il vero centro geometrico della cometa (*Ground Truth*). Lo *spread* quantifica invece la stabilità del metodo attraverso la deviazione standard dei risultati: nel test di zoom indica l'oscillazione al variare del ritaglio, mentre nel test di offset valuta la robustezza a fronte di un puntamento iniziale impreciso da parte dell'operatore. Lo *Score Definitivo* sintetizza queste componenti, offrendo un indice numerico della validità globale di ogni modello.

Dall'osservazione dei dati complessivi emerge un dettaglio fondamentale: il miglior algoritmo in classifica commette un errore medio residuo compreso tra i 2.4 e i 4.7 pixel rispetto al *Ground Truth*. Sebbene a una prima lettura possa sembrare uno scostamento rilevante, si tratta in realtà di un valore ottimo, diretta conseguenza della severità con cui le immagini sintetiche sono state

appositamente costruite. Il dataset è stato infatti progettato come uno *stress-test* estremo, dove l'emissione anisotropa del gas e il forte angolo di fase solare creano un fotocentro apparente molto distante dal vero nucleo roccioso, il quale rimane otticamente invisibile. Di fronte a questa complessa illusione fisica, i metodi tradizionali si lasciano inesorabilmente ingannare, convergendo banalmente sul punto più luminoso o venendo trascinati dalla coda, accumulando errori di decine di pixel. Al contrario, i risultati dimostrano come i modelli matematici più avanzati riescano a decodificare l'ingannevole distribuzione di luce, compensando attivamente lo sbilanciamento della chioma e arrivando molto più vicini al centro reale seppur nascosto.

L'analisi dei risultati evidenzia differenze significative tra le varie famiglie di algoritmi, facendo emergere un chiaro compromesso tra l'accuratezza assoluta e la resilienza operativa. Il *Modello Asimmetrico a Residui Pesati* registra l'errore metrico inferiore (2.45 pixel), dimostrando la massima efficacia nel mitigare il bias morfologico indotto dalla chioma. Tuttavia, questo approccio presenta una dispersione statistica (*spread*) più marcata rispetto al suo diretto concorrente. Di contro, il *Profilo Asimmetrico a Quadranti* emerge come il metodo più bilanciato e robusto dell'intero *benchmark*: pur registrando un errore medio lievemente superiore (4.62 pixel), garantisce un'elevata stabilità, mantenendo le oscillazioni al di sotto di 0.4 pixel al variare delle dimensioni della finestra di interesse o delle coordinate di innesco. Tale invarianza rende questo algoritmo lo strumento più idoneo per garantire la ripetibilità della misurazione scientifica.

Un comportamento antitetico si osserva nei modelli a simmetria radiale, , quali il *Profilo Radiale Addolcito (Vincolato)* e il *Profilo Radiale con Addolcimento Centrale*. Sebbene registrino un errore assoluto maggiore (causato dall'intrinseca incapacità di modellare l'asimmetria dell'emissione gassosa), essi offrono una stabilità pressoché totale, con fluttuazioni inferiori a 0.2 pixel. Il vincolo geometrico della simmetria forza infatti la convergenza del modello sul picco fotometrico, rendendo l'algoritmo largamente insensibile alle irregolarità periferiche della distribuzione luminosa.

Alla luce dei dati sperimentali raccolti, l'applicativo adotta il *Profilo Asimmetrico a Quadranti* come motore di *fit* primario. Tale scelta è motivata dalla superiore resilienza e stabilità dei risultati offerta da questo modello, caratteristiche ritenute prioritarie rispetto alla precisione fotometrica pura per garantire misurazioni astrometriche consistenti e indipendenti dall'errore umano. Per garantire la continuità operativa dell'applicativo, è stata implementata una gerarchia di protezione a due livelli. Qualora non si riesca a raggiungere la convergenza entro i limiti di iterazione prefissati o il risolutore restituisca un errore numerico, il sistema intercetta l'evento e attiva automaticamente un meccanismo di *fallback* verso il *Baricentro Luminoso Globale*. Quest'ultimo, pur essendo meno accurato nella localizzazione del nucleo, garantisce sempre una risposta deterministica, impedendo il blocco del flusso di analisi. L'implementazione software di questa logica di controllo è riportata nella Figura 4.5.

```
public Point2D FindAsymmetricQuadrantCenter(Mat region)
{
    if (region == null || region.Empty()) return new Point2D(-1, -1);

    int w = region.Width, h = region.Height;

    // 1. Preparazione dati e calcolo statistiche per stima iniziale
    double[] zArray = ExtractDoubleArray(region);
    double bg = CalculatePercentile(zArray, 0.05);
    double amp = (zArray.Max() - bg) * 1.5;

    // 2. Parametri iniziali: [xc, yc, A, nx+, nx-, ny+, ny-, r0, bg]
    double[] p0 = { w/2.0, h/2.0, amp, 1.0, 1.0, 1.0, 1.0, 1.5, bg };
    double[] bndl = { 0, 0, 0, 0.1, 0.1, 0.1, 0.1, 0.1, -1e6 };
    double[] bndu = { w, h, 1e9, 5.0, 5.0, 5.0, 5.0, 20.0, 1e9 };

    try
    {
        // 3. Configurazione risolutore ALGLIB per fit non lineare
        alglib.lsfitstate state;
        alglib.lsfitreport rep;
        double[,] xyData = GetCoordinatesGrid(w, h);

        alglib.lsfitcreatef(xyData, zArray, p0, 1e-5, out state);
        alglib.lsfitsetbc(state, bndl, bndu);

        alglib.lsfitfit(state, (double[] p, double[] x, ref double
            func, object obj) =>
        {
            double dx = x[0] - p[0], dy = x[1] - p[1];
            double term_x = Math.Pow(Math.Abs(dx), dx >= 0 ? p[3] :
                p[4]);
            double term_y = Math.Pow(Math.Abs(dy), dy >= 0 ? p[5] :
                p[6]);
            func = (p[2] / (term_x + term_y + p[7])) + p[8];
        }, null, null);

        alglib.lsfitresults(state, out int info, out double[]
            finalParams, out rep);

        // 4. Gestione fallimento convergenza numerica (Fallback 1)
        if (info <= 0) return FindCentroid(region, sigma: 2.0);

        return new Point2D(finalParams[0], finalParams[1]);
    }
    catch
    {
        // 5. Gestione eccezioni generiche (Fallback 2)
        return FindCentroid(region, sigma: 2.0);
    }
}
```

Figura 4.5. Implementazione in linguaggio C# del *Profilo Asimmetrico a Quadranti*. Si noti l'integrazione dei vincoli fisici e la logica di protezione che attiva il baricentro in caso di mancata convergenza numerica.

4.4 Infrastruttura Astrometrica: WCS e Predizione Orbitale

Nelle sezioni precedenti sono stati esaminati i motori matematici che permettono di allineare immagini basandosi sul calcolo dei centroidi luminosi. Tuttavia, per un software dedicato allo studio morfologico e scientifico delle comete, la pura geometria dei pixel non è sufficiente. Come già sottolineato, a differenza delle stelle, che possono essere approssimate a sorgenti puntiformi fisse, una cometa è un oggetto intrinsecamente "vivo", diffuso e dinamicamente erratico, caratterizzato da un moto proprio indipendente rispetto alla volta celeste. Per isolare strutture a bassissimo contrasto come getti, ventagli o disconnessioni della coda, è fondamentale ancorare le immagini non solo a un sistema di riferimento interno, ma alle reali coordinate dell'universo.

Questa sezione esplora l'infrastruttura astrometrica che precede e guida l'analisi visiva. Il processo si fonda su due pilastri tecnologici essenziali:

1. **La calibrazione spaziale dell'immagine** (comunemente nota come *Plate Solving*), che permette di generare un WCS (*World Coordinate System*) e tradurre ogni singolo pixel in coordinate equatoriali precise.
2. **La predizione dinamica**, garantita dai modelli matematici del JPL (*Jet Propulsion Laboratory*) della NASA, che forniscono le effemeridi ad alta precisione del corpo celeste.

L'implementazione di questa infrastruttura costituisce un sostanziale salto di qualità per la *pipeline* di elaborazione. Essa introduce infatti un netto miglioramento metodologico poiché permette di sfruttare appieno il ricco patrimonio di metadati integrati nativamente nell'*header* del formato FITS.

La combinazione di questi due strumenti permette al software di sapere esattamente "dove sta guardando" il telescopio e "dove dovrebbe trovarsi" la cometa. Di seguito verrà analizzato come il software integri servizi di risoluzione esterni e dati orbitali per eseguire un allineamento astrometrico primario, definendo al contempo i margini di errore fisiologici che rendono indispensabili i successivi algoritmi di raffinamento sub-pixel.

4.4.1 Il Plate Solving: Dalla Matrice di Pixel alla Sfera Celeste

Quando un telescopio acquisisce l'immagine di una cometa, il file grezzo prodotto dal sensore è, da un punto di vista puramente informatico, soltanto una griglia bidimensionale anonima. È una matrice in cui ogni cella possiede una coordinata cartesiana (x, y) e un valore di intensità luminosa, ma è del tutto priva di contesto spaziale. Per poter condurre un'analisi morfologica e dinamica rigorosa, è indispensabile ancorare questa griglia astratta al sistema di riferimento inerziale dell'universo. Questa delicata e fondamentale transizione prende il nome di *Plate Solving* (risoluzione astrometrica del campo inquadrato).

Il *Plate Solving* è la tecnica computazionale che si fa carico di esaminare l'immagine e identificare una corrispondenza univoca tra il *pattern* di stelle rilevato dal sensore e le mappe stellari contenute in immensi cataloghi astronomici di riferimento, come il catalogo Gaia [36]. La genialità degli algoritmi moderni risiede nel fatto che non tentano un laborioso confronto stella per stella, operazione che fallirebbe al minimo errore di scala o rotazione. Al contrario, utilizzano un approccio matematico noto come *hashing geometrico*.

Più nello specifico, il sistema estrae le coordinate dei punti più luminosi dell'immagine e li raggruppa idealmente a formare dei poligoni, tipicamente quaterne o tetraedri [37]. Per ognuna di queste figure calcola un "hash", ovvero un codice identificativo univoco basato esclusivamente sui rapporti tra le distanze e gli angoli interni delle stelle che compongono il poligono. La potenza di questa strategia risiede nella sua invarianza: un quadrato o un rombo manterranno lo stesso hash indipendentemente da quanto l'immagine venga ingrandita, ruotata o traslata.

L'efficienza computazionale di questo processo dipende drasticamente dalle informazioni fornite in ingresso. Se il motore di risoluzione dovesse cercare questi hash geometrici scandagliando l'intera volta celeste e testando ogni possibile ingrandimento, l'operazione richiederebbe svariati

minuti (operazione nota come *Blind Solving* puro). Per abbattere i tempi di calcolo a frazioni di secondo, il software sviluppato sfrutta i metadati approssimativi già presenti o calcolati dall'utente, come la lunghezza focale del telescopio e la dimensione fisica dei pixel del sensore. Questi parametri definiscono la scala dell'immagine in arcosecondi per pixel, un "suggerimento" (*hint*) cruciale che restringe il campo di ricerca da miliardi di possibili combinazioni a un intorno limitato di pochi gradi quadrati attorno alle coordinate di puntamento previste (*Local Solving*).

4.4.2 Lo Standard WCS e la Codifica FITS

Il traguardo finale del *Plate Solving* non è semplicemente la consapevolezza visiva di cosa si stia osservando, ma la generazione di una complessa impalcatura matematica nota come WCS. Una volta trovata la corrispondenza esatta, il risolutore inietta nell'header del file FITS una serie di chiavi standardizzate che definiscono la matrice di trasformazione tra il piano del sensore e la sfera celeste [38].

I parametri fondamentali sono costituiti dalle chiavi `CRPIX1` e `CRPIX2`, che stabiliscono un punto di ancoraggio arbitrario sul sensore (generalmente coincidente con il pixel centrale), e dalle corrispondenti `CRVAL1` e `CRVAL2`, che associano a tale punto le esatte coordinate equatoriali di Ascensione Retta e Declinazione. Per descrivere la proiezione della volta celeste sull'intera immagine, il sistema compila le chiavi relative alla matrice di trasformazione lineare (tipicamente `CD1_1`, `CD1_2`, `CD2_1` e `CD2_2`), in accordo con le convenzioni standard per le proiezioni sferiche [39]. Tali valori codificano simultaneamente l'esatta scala dell'immagine e l'angolo di rotazione del sensore rispetto al Nord Celeste.

Nelle soluzioni astrometriche di maggiore precisione, a queste informazioni si affiancano le chiavi del protocollo SIP (*Simple Imaging Polynomial*), tra cui `A_ORDER`, `B_ORDER` e i rispettivi coefficienti polinomiali `A_p_q` e `B_p_q` [40]. Questi parametri descrivono analiticamente le distorsioni ottiche non lineari del sistema (quali le aberrazioni a barile o a cuscinetto), consentendo al software di derivare coordinate con accuratezza sub-pixel anche nelle regioni periferiche del campo di vista.

Per lo studio specifico delle comete, l'implementazione del WCS rappresenta un prerequisito fondamentale. La definizione della scala esatta consente di convertire le distanze misurate in pixel in chilometri fisici, passaggio obbligato per l'analisi morfologica della chioma e della coda. Parallelamente, la determinazione rigorosa dell'orientamento spaziale permette di correlare la direzione delle strutture osservate con i vettori del vento solare. Infine, la correzione delle distorsioni ottiche assicura che, durante i processi di *stacking*, l'allineamento delle stelle di campo risulti perfetto in ogni sua parte, prevenendo la formazione di artefatti che comprometterebbero l'integrità fotometrica del segnale scientifico.

4.4.3 Il Motore di Risoluzione: Integrazione Locale di ASTAP

Per tradurre in pratica i complessi calcoli del *Plate Solving* e la generazione dello standard WCS, il software necessita di un motore computazionale che sia contemporaneamente robusto e fulmineo. Invece di dipendere da lenti servizi cloud o Application Programming Interface (API) remote che richiederebbero il caricamento online di pesanti file grezzi, il programma integra e gestisce nativamente l'utilizzo di ASTAP. Sviluppato dall'astronomo Han Kleijn, ASTAP è universalmente riconosciuto come l'attuale standard di eccellenza per l'astrometria locale.

La particolarità architettonica del software risiede nell'interfacciamento diretto e invisibile con l'eseguibile di ASTAP già installato sul computer dell'utente. Durante il flusso di lavoro, il programma non costringe mai l'operatore a uscire dall'interfaccia principale per risolvere le immagini manualmente. Al contrario, il software estrae autonomamente i metadati preliminari, calcola la scala dell'immagine in arcosecondi per pixel e invia questi parametri ad ASTAP eseguendolo in *background* tramite processi di sistema [41]. Ricevendo questi *hint* geometrici ed equatoriali precisi, ASTAP evita il dispendioso processo di ricerca alla cieca su tutta la volta celeste, restringendo l'analisi a un raggio locale strettissimo.

Il risultato è un *Plate Solving* che si conclude tipicamente in frazioni di secondo per ogni singola immagine. ASTAP calcola la matrice di trasformazione, inietta le chiavi WCS direttamente nell'header del file FITS e restituisce il controllo al programma principale in modo del tutto trasparente. Questa integrazione totalmente *offline* e automatizzata è fondamentale per l'analisi cometaria: permette infatti di processare in blocco sequenze temporali composte da decine o centinaia di scatti ad alta risoluzione senza subire colli di bottiglia dovuti alla connessione internet, garantendo un flusso di lavoro astrometrico fluido e ininterrotto.

4.4.4 Modellazione Orbitale: JPL e Gruppo *Solar System Dynamics*

Per ottenere la precisione necessaria a un allineamento sub-pixel, il software si interfaccia con l'autorità mondiale in materia di meccanica celeste: il JPL della NASA. Il *Solar System Dynamics* (SSD) Group è l'ente responsabile della definizione ufficiale delle orbite di tutti i corpi del Sistema Solare, basandosi su integrazioni numeriche complesse delle equazioni del moto per un sistema a n-corpi [42].

Questi modelli, noti come *Development Ephemerides* (DE), tengono conto delle perturbazioni planetarie, delle interazioni relativistiche e dell'influenza gravitazionale aggregata dei corpi minori. Lo standard di riferimento utilizzato nel software è la serie DE440/DE441 [43], che rappresenta lo stato dell'arte nella precisione orbitale, integrando i dati più recenti provenienti dalle missioni interplanetarie.

L'interrogazione di questi dati avviene tramite il sistema JPL Horizons, un motore di calcolo dinamico accessibile via API [42].

4.4.5 Limiti Operativi: Forze Non Gravitazionali ed Errori Sistematici

Tuttavia, anche disponendo della potenza di calcolo del JPL, la predizione della posizione di una cometa rimane intrinsecamente più complessa e incerta rispetto a quella di un asteroide o di un pianeta a causa della sua stessa natura fisica [44]. Le comete sono oggetti attivi: avvicinandosi al perielio, il calore solare sublima i ghiacci volatili presenti sulla superficie o appena sotto la crosta del nucleo. L'espulsione violenta di getti di gas e polveri agisce come un vero e proprio motore a razzo naturale, generando una spinta che devia la cometa dalla sua orbita puramente gravitazionale.

Il JPL modella tipicamente questo fenomeno utilizzando il formalismo standard di Marsden (1973) [45], che scompone questa accelerazione anomala in tre vettori componenti diretti rispettivamente in senso radiale rispetto al Sole, trasversale al piano orbitale e normale ad esso. Nonostante la raffinatezza di questo modello, l'attività di una cometa è spesso stocastica e imprevedibile, soggetta a *outburst* improvvisi o all'attivazione asimmetrica di nuovi getti. Di conseguenza, l'orbita teorica possiede un margine di incertezza intrinseco che rende inaffidabile un allineamento basato esclusivamente sui dati effemeridei e sulla mappa stellare (*Plate Solving*). Questi margini di incertezza si sommano, definendo un limite fisico e strumentale ben preciso articolato in tre fattori ineliminabili:

1. **Errore Astrometrico e Strumentale:** la soluzione WCS trascritta nell'header FITS non è mai assoluta. Essa risente della turbolenza atmosferica (*seeing*), del campionamento del sensore e delle microscopiche distorsioni ottiche residue. Come evidenziato in letteratura [46], il limite teorico di precisione per il calcolo del centroide su un sensore digitale è intrinsecamente limitato dalla PSF e dalla diffusione di carica sui pixel, specie in condizioni di sottocampionamento rispetto al limite di Nyquist. Nella pratica astrometrica cometaria, le incertezze di base spingono i ricercatori ad assegnare un errore strumentale di 1 pixel se la condensazione appare stellare, oppure di 2 pixel nei casi in cui l'immagine risulti diffusa a causa del *seeing* o di errori di inseguimento [44]. A questo si aggiunge la precisione temporale: un orologio di sistema del PC acquisitore fuori sincrono anche solo di frazioni di secondo introduce un immediato errore di traslazione sulla volta celeste.

2. **Errore Dinamico delle Effemeridi:** i modelli del JPL rimangono approssimazioni matematiche di una realtà caotica. Per comete periodiche storiche e ben documentate lo scostamento è minimo. Tuttavia, per comete iperattive l'imprevedibilità dei getti rende il modello classico di Marsden insufficiente sull'intero arco orbitale senza continue ricalibrazioni. Se non vengono fornite osservazioni astrometriche recenti per aggiornare i parametri, la discrepanza tra la traiettoria reale e quella calcolata cresce in modo inaccettabile, richiedendo l'adozione di modelli più complessi (come il *Rotating Jet Model*) per compensare gli errori dinamici accumulati [44].

3. **Errore Fisico Sistematico (Centro di Massa vs Optocentro):** le equazioni del JPL calcolano la posizione del baricentro gravitazionale (il nucleo solido e oscuro), ma il sensore fotografico registra l'optocentro (il picco di luce della chioma e delle polveri). L'asimmetria della chioma cometaria introduce un evidente *tailward bias*, un offset sistematico verso la coda che varia al variare dell'apertura sintetica utilizzata per la misurazione [44]. Studi sull'astrometria di precisione hanno dimostrato che questo offset costringe i ricercatori ad applicare correzioni estrapolate fino a $2.0''$ per individuare la vera posizione del nucleo. Inoltre, misurazioni non filtrate accuratamente possono presentare deviazioni che arrivano fino a $5.0''$ rispetto all'orbita reale [44].

Sommando queste tre componenti, l'errore totale di puntamento si attesta generalmente in un intervallo compreso tra 1.5 e 5.0 arcosecondi. Traducendo questo valore angolare nella griglia fisica del sensore (considerando un campionamento tipico di $1.0''$ o $2.0''$ per pixel), risulta evidente che l'uso esclusivo di JPL e WCS non permetterà mai di raggiungere una precisione sub-pixel [46]. L'errore effettivo consisterà in uno scostamento di pochi pixel nei casi più fortunati, per arrivare ad alcune decine di pixel in presenza di focali molto lunghe. In definitiva, questo approccio teorico garantisce un inquadramento preliminare eccellente e infallibile, ma rende imperativo l'intervento dei successivi algoritmi di analisi visiva per centrare l'immagine.

4.5 Allineamento sul Campo Stellare (Registrazione Siderale)

Fino a questo punto, l'analisi si è concentrata sulle metodologie necessarie a individuare l'optocentro della cometa, operazione fondamentale per allineare le sequenze sul nucleo in movimento. Tuttavia, per un'elaborazione scientifica ed estetica completa, è altrettanto cruciale essere in grado di rilevare e tracciare con precisione il *pattern* stellare. L'allineamento delle immagini rispetto al campo stellare, tecnicamente definito registrazione siderale, costituisce una procedura cardine. La necessità di questa operazione nasce dalla natura dinamica del soggetto: una cometa si muove rispetto allo sfondo fisso delle stelle, creando una discrepanza inevitabile in fase di acquisizione dove, a seconda del sistema di inseguimento scelto, uno dei due elementi risulterà mosso.

Allineare la sequenza di scatti sulle stelle permette di fissare matematicamente il sistema di riferimento inerziale dell'universo, "congelando" la posizione degli astri e trasformando la cometa in un oggetto in transito. Questa operazione è il prerequisito indispensabile per tecniche avanzate come la rimozione delle stelle, che consente di isolare e analizzare le strutture deboli della chioma e della coda senza la contaminazione luminosa del tappeto stellare di fondo, oltre a essere fondamentale per misurazioni astrometriche precise che richiedono una griglia di coordinate fisse e note.

4.5.1 Implementazione Algoritmica per la Registrazione Siderale

All'interno del software sviluppato, l'algoritmo implementato si basa su una combinazione di filtraggio morfologico e correlazione di fase nel dominio delle frequenze.

Il processo logico inizia con una fase di pre-elaborazione cruciale. Invece di confrontare le immagini grezze, che potrebbero contenere gradienti luminosi o la nebulosità diffusa della cometa

stessa (elementi che disturberebbero l'allineamento stellare), il software applica un filtro matematico (come l'operatore di Sobel) per estrarre la magnitudine dei gradienti. Questa operazione trasforma l'immagine in una mappa che evidenzia esclusivamente le transizioni nette e i bordi ad alta frequenza, isolando di fatto le stelle puntiformi e scartando tutto ciò che risulta sfocato o diffuso, come appunto la cometa o l'inquinamento luminoso del fondo cielo.

Una volta ottenuta questa mappa dei bordi, l'algoritmo non cerca di calcolare lo spostamento sull'immagine intera in un unico passaggio, ma adotta una strategia *divide et impera* per massimizzare l'affidabilità. L'area di lavoro viene suddivisa in una griglia e per ogni settore viene eseguito un calcolo indipendente dello spostamento. Il frammento di codice C# che gestisce questa logica iterativa e il calcolo della correlazione è riportato in Figura 4.6.

```
public Point2D ComputeStarFieldShift(Mat reference, Mat target)
{
    // 1. Estrazione dei bordi ad alta frequenza (Filtro di Sobel)
    using var refEdge = ApplySobelFilter(reference);
    using var tgtEdge = ApplySobelFilter(target);

    // 2. Setup griglia 3x3 e finestra di Hanning (Anti-aliasing)
    int cellW = refEdge.Width / 3, cellH = refEdge.Height / 3;
    List<Point2d> shifts = new List<Point2d>();

    using var hanningWin = new Mat();
    Cv2.CreateHanningWindow(hanningWin, new Size(cellW, cellH),
        MatType.CV_32FC1);

    // 3. Calcolo indipendente dello spostamento per settore
    for (int r = 0; r < 3; r++) {
        for (int c = 0; c < 3; c++) {

            var rect = new Rect(c * cellW, r * cellH, cellW, cellH);
            using var cellRef = new Mat(refEdge, rect);
            using var cellTgt = new Mat(tgtEdge, rect);

            // [...] Omessi controlli di cella vuota e conversioni a
                32-bit

            // Correlazione di fase nel dominio delle frequenze (FFT)
            Point2d shift = Cv2.PhaseCorrelate(cellRef, cellTgt,
                hanningWin, out double response);

            // Filtro di confidenza sui risultati locali
            if (response > 0.05) shifts.Add(shift);
        }
    }

    // 4. Clustering per reiezione outlier e calcolo dello shift globale
    Point2d finalShift = shifts.Count > 0 ?
        ComputeConsensusShift(shifts) : new Point2d(0,0);

    return new Point2D(finalShift.X, finalShift.Y);
}
```

Figura 4.6. Implementazione compatta dell'algoritmo di registrazione siderale, illustrante la scomposizione a griglia e l'impiego della correlazione di fase FFT per il calcolo dei vettori sub-pixel.

4.6 Architettura Software: Astrometria e Dinamica Orbitale

In questa sezione si procede ad illustrare l'architettura effettiva implementata nel software per la gestione delle funzionalità appena discusse. Il codice è stato costruito seguendo una rigorosa separazione delle responsabilità, fondendo servizi web aerospaziali, calcoli matematici non lineari, gestione della memoria e rendering reattivo. Il flusso dei dati viaggia attraverso livelli di astrazione decrescenti, partendo dall'interrogazione di database astronomici per arrivare fino alla manipolazione del singolo pixel sullo schermo.

Al vertice di questa piramide decisionale si trova l'*AlignmentCoordinator*, il cui compito è orchestrare l'intera sequenza di operazioni; infatti, il programma si rivolge esclusivamente a questa classe quando è necessario effettuare un'allineamento. Per questo motivo, il coordinatore non esegue calcoli matematici complessi né sposta fisicamente i pixel, ma ha il compito di dirigere i vari servizi specializzati, stabilendo cosa fare e in che ordine. Inizialmente, il sistema recupera i metadati leggendo l'header dei file FITS per identificare rapidamente informazioni essenziali come la data di scatto, le coordinate dell'osservatorio, la risoluzione dell'immagine e la presenza di calibrazioni spaziali (WCS). Fatto ciò, delega il calcolo dei nuclei identificati nelle immagini per creare una mappa delle posizioni. Infine, l'elaborazione si conclude con la fase di esecuzione e salvataggio: basandosi sulla mappa appena creata, la classe incarica altri servizi di eseguire il "lavoro pesante", che consiste nello spostare materialmente i pixel, ridimensionare le immagini, registrare lo storico delle operazioni (HISTORY) negli header e salvare i file finali allineati su disco.

Sulla base di questi dati preliminari, il sistema utilizza l'*AlignmentService* per le operazioni matematiche e geometriche. Questa classe sfrutta il pattern di sviluppo *Strategy* [9] per decidere dinamicamente quali elaborazioni applicare a un'immagine, in base al target e alla modalità scelta dall'utente. Nello specifico, per il centraggio sul nucleo della cometa sono previste tre opzioni:

- **Modalità automatica**: cerca di stabilire da sola la posizione del nucleo all'interno dell'immagine. Funziona bene per comete evidenti, ma può risultare insufficiente per bersagli nascosti o per sequenze con forti imprecisioni di inseguimento.
- **Modalità guidata**: richiede che l'utente selezioni la posizione indicativa della cometa soltanto nella prima e nell'ultima immagine; in questo modo, il software calcola la traiettoria interpolando la posizione presunta nei fotogrammi intermedi, evitando un inserimento manuale continuo.
- **Modalità manuale**: offre il pieno controllo sull'area analizzata, permettendo di trattare correttamente anche comete molto piccole rispetto al campo inquadrato.

Invece, per l'allineamento sul pattern stellare le modalità sono due:

- **Automatica**: risolve con successo la maggior parte delle casistiche tramite la *Phase Correlation* nel dominio della frequenza.
- **Manuale**: mantenuta come soluzione di riserva per gestire errori imprevisti.

Per tutte le modalità, è sempre possibile aggiustare manualmente i punti individuati prima di procedere.

La traduzione di questa complessa geometria in immagini finali è affidata al *BatchProcessingService*. Questo motore modifica i file FITS applicando l'effettiva traslazione sub-pixel (*Warping*) e aggiornando contestualmente le matrici WCS affinché l'immagine, che sia ritagliata o espansa, mantenga la sua validità astronomica. Durante l'esportazione, una funzione di *Smart Promotion* analizza i valori dei pixel, innalzando automaticamente la profondità di bit (ad esempio verso *Float* o *Double*) se le operazioni matematiche hanno generato valori decimali, prevenendo così qualsiasi perdita di informazione scientifica. Al termine di ogni fotogramma, il sistema forza spietatamente il *Garbage Collector* a deframmentare la Large Object Heap (LOH), salvaguardando il software da inevitabili crash causati dalla frammentazione della memoria. Questa classe è stata progettata

in modo indipendente dalle logiche di allineamento e viene infatti utilizzata trasversalmente dal programma per tutte quelle operazioni che richiedono un'elaborazione parallela su intere sequenze di immagini.

Per ciascuna modalità di allineamento è possibile sfruttare i dati WCS già presenti negli header FITS. In particolare, se il bersaglio dell'allineamento è una cometa e l'utente lo richiede, il coordinatore è in grado di connettersi ai server della NASA attraverso la classe *JplHorizonsService*. Questo modulo non si limita a inviare una semplice richiesta, ma implementa una resiliente strategia di fallback a cascata: tenta prima la ricerca per designazione standard, poi passa ai nomi testuali e, nel caso in cui i server restituiscano un'ambiguità su oggetti multipli, filtra i candidati analizzando l'anno dell'osservazione, estraendo infine le esatte coordinate celesti (Ascensione Retta (AR) e Declinazione (DEC)). Se invece il target è un campo stellare, il sistema sfrutta direttamente i dati celesti di riferimento del primo fotogramma.

```
private Point2D SolveTpvInverseNewton(double targetXi, double targetEta)
{
    double xi = targetXi, eta = targetEta;
    const int maxIter = 20;
    const double tolerance = 1e-9;

    for (int i = 0; i < maxIter; i++)
    {
        double r = Math.Sqrt(xi * xi + eta * eta);

        // 1. Calcolo dei residui rispetto al bersaglio
        double f = ComputePvSum(1, xi, eta, r) - targetXi;
        double g = ComputePvSum(2, xi, eta, r) - targetEta;

        // Condizione di uscita: convergenza sub-pixel raggiunta
        if (Math.Abs(f) < tolerance && Math.Abs(g) < tolerance) break;

        // 2. Calcolo della Matrice Jacobiana (derivate parziali)
        var d1 = ComputePvDerivs(1, xi, eta, r);
        var d2 = ComputePvDerivs(2, xi, eta, r);
        double det = (d1.dXi * d2.dEta) - (d1.dEta * d2.dXi);

        if (Math.Abs(det) < 1e-15) { xi -= f; eta -= g; continue; }

        // 3. Aggiornamento delle coordinate (Metodo di Newton-Raphson
            2D)
        xi -= (d2.dEta * f - d1.dEta * g) / det;
        eta -= (-d2.dXi * f + d1.dXi * g) / det;
    }

    return new Point2D(xi, eta);
}
```

Figura 4.7. Risoluzione iterativa delle distorsioni ottiche TPV tramite il metodo numerico di Newton-Raphson bidimensionale, con calcolo dello Jacobiano.

Una volta ottenute le coordinate celesti, è indispensabile capire a quale punto esatto del sensore fotografico corrispondano. In questa fase entra in gioco la classe *WcsTransformation*. Questo motore matematico converte i gradi celesti in coordinate fisiche di pixel (e viceversa), ma non si limita a una conversione basilare (come se il cielo fosse una mappa perfettamente piatta): è stato infatti progettato per correggere le inevitabili distorsioni ottiche introdotte dalle lenti e dagli specchi dei telescopi. Per riuscirci, il software esegue prima dei calcoli di trigonometria sferica, "appiattendo" la volta celeste su un piano ideale a due dimensioni. Successivamente, per compensare le deformazioni ottiche e raddrizzare l'immagine, applica algoritmi matematici

avanzati (come il metodo di Newton-Raphson, di cui è fornita un'implementazione in Figura 4.7, o l'iterazione di punto fisso) capaci di decodificare i complessi formati di distorsione standard usati in astronomia (noti come TPV e Simple Imaging Polynomial (SIP)).

Tuttavia, poiché tali dati non sono sempre nativamente presenti, è stata sviluppata una funzionalità parallela all'allineamento per eseguire la risoluzione delle immagini, gestita dal *PlateSolvingCoordinator* e dal *PlateSolvingService*. Il processo inizia con una diagnosi preventiva in cui il servizio verifica la presenza dei prerequisiti fondamentali per la risoluzione astrometrica, come la lunghezza focale e la dimensione dei pixel. Per garantire la massima sicurezza dei dati originali, il sistema genera dinamicamente una versione temporanea delle immagini: se l'immagine ha subìto modifiche in RAM, crea un file FITS assemblando l'header aggiornato con i pixel estratti in modo sicuro dall'HDU corretto; altrimenti, realizza una copia fisica su disco. Il motore avvia quindi l'eseguibile esterno ASTAP in modo asincrono, iniettando parametri intelligenti (*hints* come le coordinate stimate e un raggio di ricerca adattivo) per accelerare la convergenza, e intercetta in tempo reale lo *standard output* per fornire un feedback visivo continuo. L'intera procedura non apre finestre aggiuntive, rimanendo completamente trasparente per l'utente. Una volta conclusa l'operazione, i parametri calcolati vengono estratti chirurgicamente (solo le chiavi WCS e i coefficienti di distorsione) e iniettati nell'header originale, preservando la storia del file e arricchendolo con tag proprietari di tracciamento (PLTSOLVD, SOLVER) e stringhe HISTORY dettagliate.

4.7 Interfaccia Utente: Risoluzione Astrometrica e Allineamento

Le operazioni di calibrazione spaziale e di registrazione geometrica dei fotogrammi rappresentano il fulcro dell'elaborazione astrometrica. Per gestire la complessità di questi calcoli, sono state sviluppate due interfacce utente dedicate, progettate per guidare l'operatore, prevenire errori di configurazione e fornire un *feedback* visivo costante durante le elaborazioni più onerose.

4.7.1 Risoluzione Astrometrica (Plate Solving)

La prima interfaccia è dedicata alla calibrazione spaziale delle immagini ed è accessibile navigando nel menù principale alla voce *Modifica* → *Risoluzione Astrometrica (ASTAP)*. Analogamente alle altre funzioni di elaborazione, questa voce risulta abilitata esclusivamente previo caricamento e selezione di un nodo contenente il dataset di interesse.

Dal punto di vista della *User Experience* (UX), la finestra è stata concepita all'insegna del minimalismo operativo. Poiché il *Plate Solving* locale tramite ASTAP viene gestito autonomamente dal motore software in *background*, all'utente non è richiesta la compilazione di complessi parametri tecnici. L'interfaccia presenta una descrizione testuale del blocco di immagini su cui si sta per operare e un unico pulsante per avviare il processo di risoluzione.

Il valore aggiunto di questa schermata risiede nel *feedback* fornito durante l'esecuzione. Il software intercetta in tempo reale lo *standard output* del processo esterno di ASTAP, lo ripulisce dalle stringhe di debug non necessarie e lo stampa in diretta all'interno di una console di log integrata. Questo permette all'astronomo di monitorare attivamente l'avanzamento, verificando la precisione della convergenza e l'eventuale insorgere di errori su fotogrammi specifici. Al termine della procedura, viene indicato chiaramente come l'header FITS di ogni immagine sia stato arricchito con le nuove chiavi WCS.

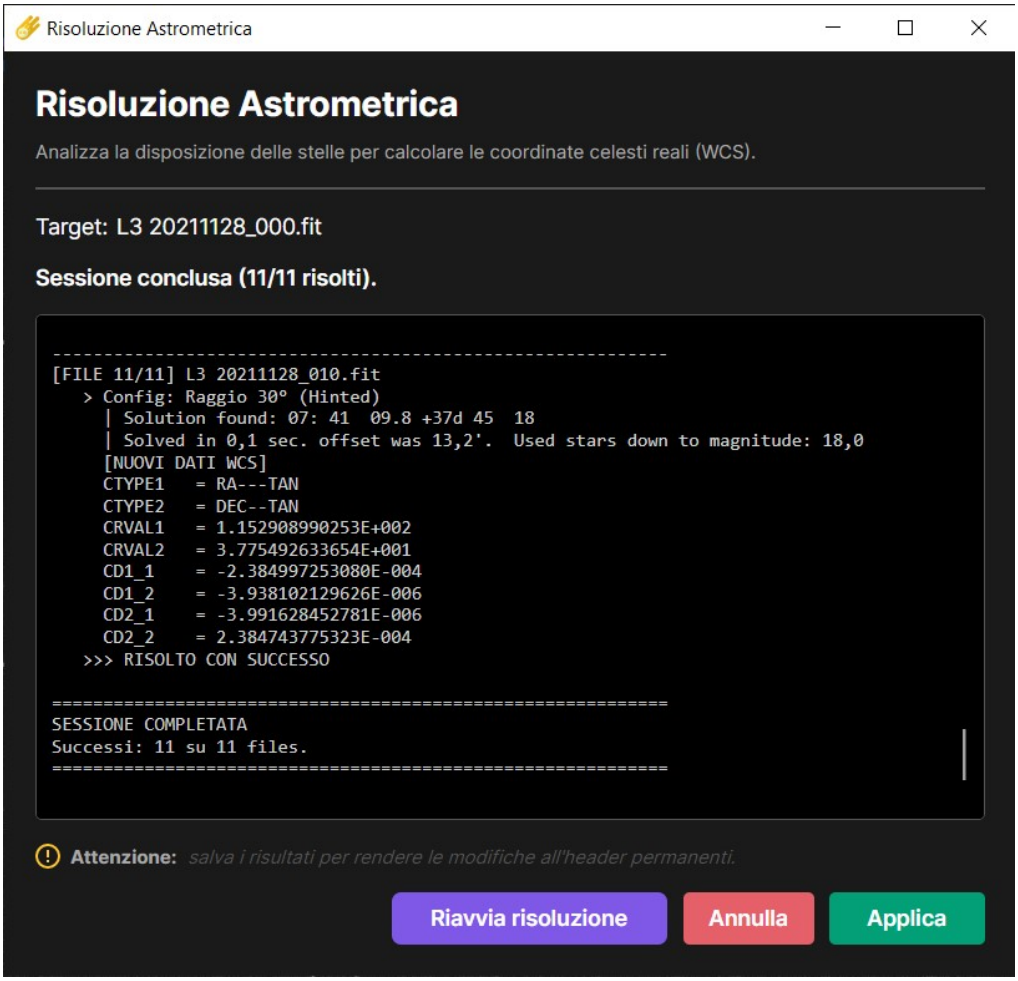

Figura 4.8. Interfaccia del modulo di risoluziona astrometrica: la console integrata mostra in tempo reale l'output di ASTAP e i metadati WCS iniettati nell'header.

4.7.2 Allineamento e Centratura

Il secondo step operativo è gestito dall'interfaccia di allineamento, accessibile tramite *Modifica* → *Allinea e Centra*. Anche in questo caso, l'accesso è subordinato alla selezione di un nodo valido nello spazio di lavoro. Una volta aperta la finestra di configurazione (visibile in Figura 4.9), l'operatore definisce il bersaglio (*Target*), scegliendo tra il nucleo della cometa o il pattern stellare. In base a questa scelta, il menù a tendina delle modalità operative si aggiorna dinamicamente.

Allineamento sulle Comete Selezionando la cometa come bersaglio, le opzioni si declinano in tre approcci:

- **Modalità Automatica:** Non richiede l'intervento manuale dell'utente. L'unica opzione configurabile riguarda l'utilizzo dei dati WCS per interrogare i server del JPL. Il sistema gestisce gli errori interrompendo l'interrogazione se l'immagine è priva di metadati, se il server è irraggiungibile o se la designazione della cometa è errata.
- **Modalità Guidata:** Introduce il parametro denominato "Raggio di Ricerca", che definisce l'area da considerare attorno al punto di stima iniziale. Graficamente, il punto indicato dall'utente appare come una croce rossa, mentre il raggio (regolabile tramite slider) è rappresentato da un *bounding box* giallo tratteggiato.

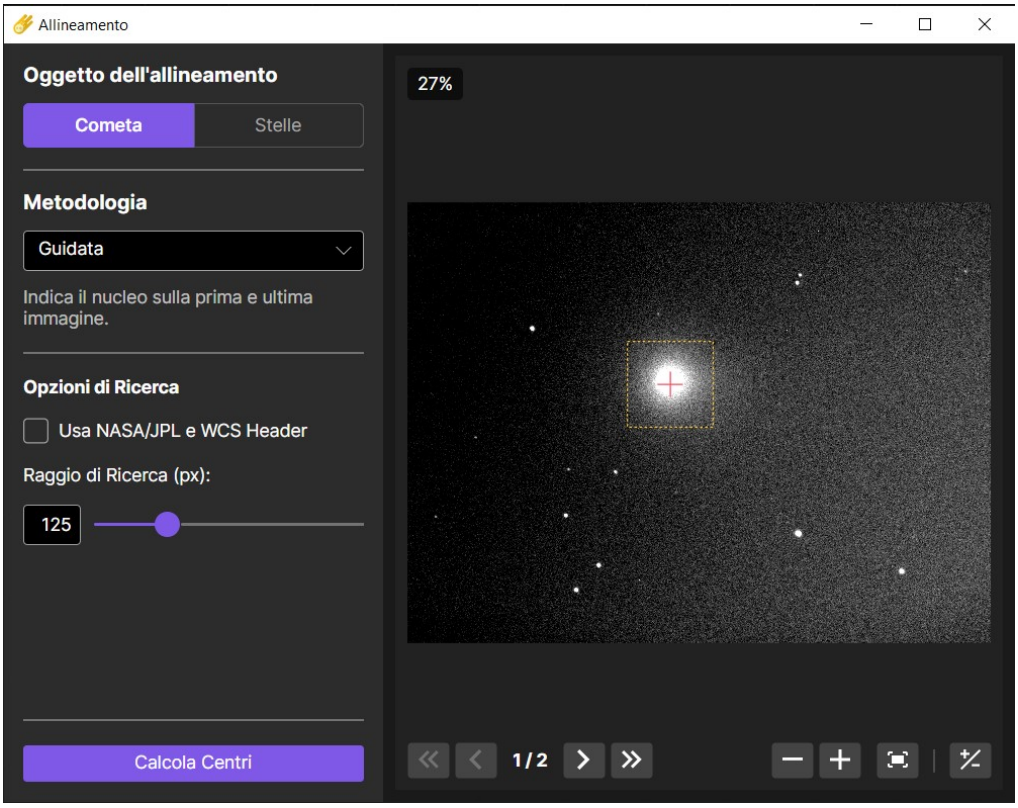

Figura 4.9. Interfaccia di configurazione dell'allineamento: selezione del bersaglio, della metodologia operativa e definizione grafica del raggio di ricerca.

- **Modalità Manuale:** Affida il controllo totale all'operatore, mantenendo unicamente il Raggio di Ricerca come parametro di raffinamento locale.

Allineamento sul Pattern Stellare Qualora il target sia il campo stellare, le opzioni si semplificano. È possibile sfruttare le matrici WCS presenti negli header; in questo caso non viene interrogato il server JPL, poiché i dati astrometrici locali sono sufficienti per calcolare la traslazione dell'inquadratura.

4.7.3 Revisione dei Centroidi ed Esecuzione Finale

Completata la configurazione, la pressione del pulsante esecutivo avvia l'elaborazione asincrona in *background*, accompagnata da un indicatore di caricamento. Al termine, l'utente viene reindirizzato alla schermata di revisione visiva (Figura 4.10).

Qui, ogni immagine mostra un marcatore grafico sul centro calcolato. L'operatore può intervenire manualmente per correggere eventuali errori di *tracking*. In questa fase è possibile scegliere se abilitare la checkbox per il ritaglio all'area comune: se abilitata, il software confina l'immagine alla sola zona d'interesse attorno alla cometa, evitando la crescita eccessiva delle dimensioni dei file e ottimizzando le performance delle analisi successive.

Una volta confermati i punti, il sistema esegue il *Warping* sub-pixel. Come mostrato in Figura 4.11, il processo restituisce una sequenza in cui il nucleo della cometa è bloccato al centro geometrico di ogni fotogramma, eliminando il moto proprio e preparando il dataset per lo *stacking*.

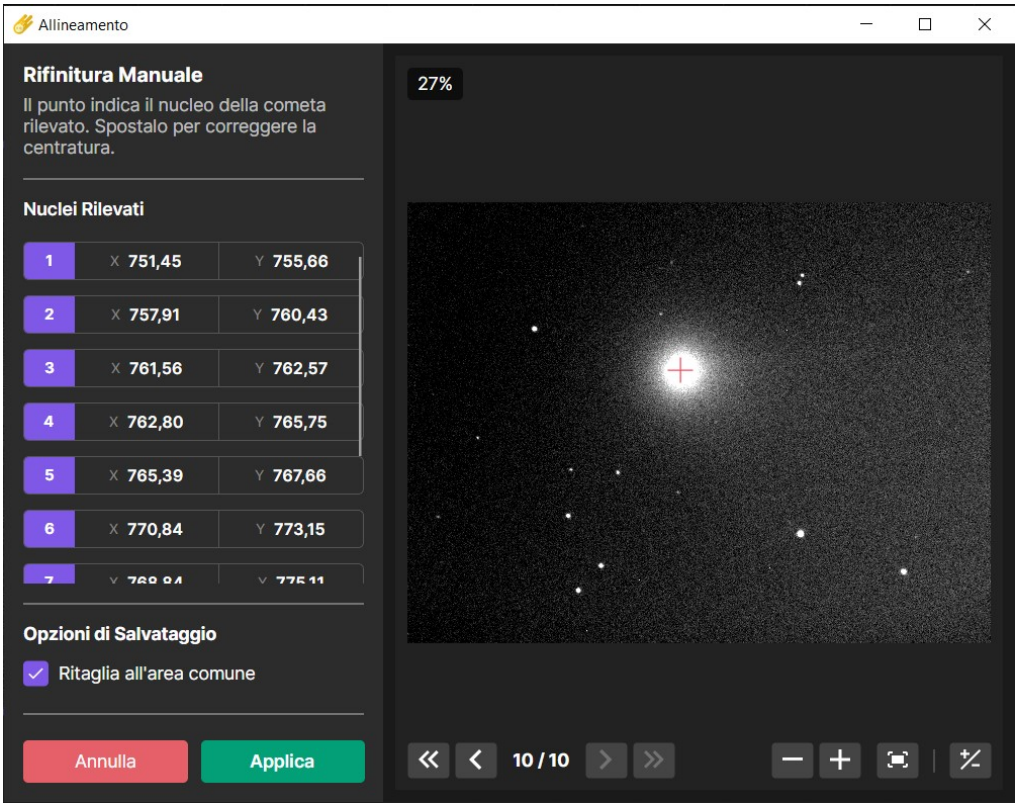

Figura 4.10. Schermata di revisione dei centroidi con opzione di cropping all'area comune.

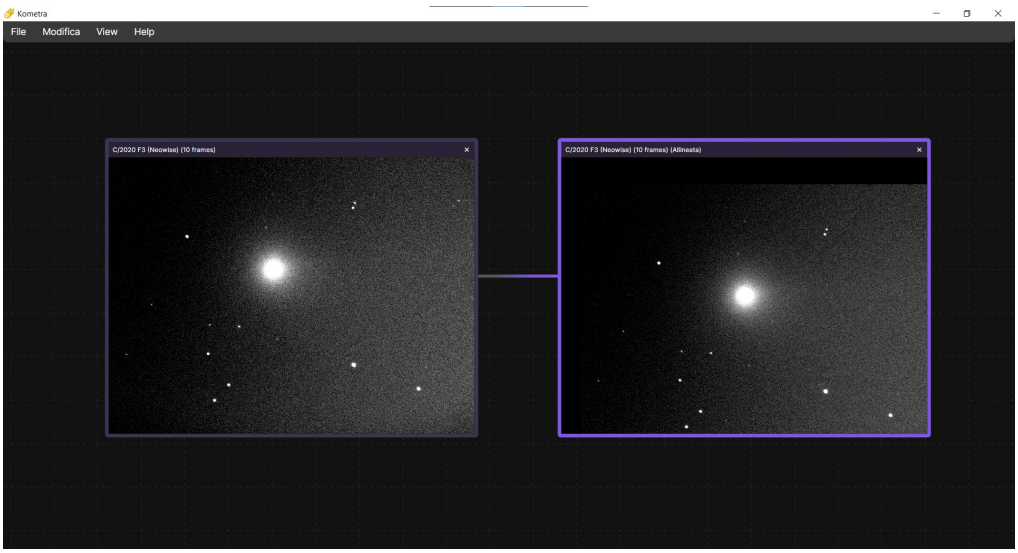

Figura 4.11. Risultato finale della centratura cometaria.

4.7.4 Operazioni di Stacking

A naturale completamento della fase di registrazione geometrica, il software mette a disposizione gli strumenti per l'integrazione fotometrica della sequenza, comunemente nota come *stacking*. Una volta che i fotogrammi sono stati allineati sul medesimo centroide, è possibile fonderli per incrementare drasticamente il rapporto segnale-rumore (SNR) dell'immagine finale.

L'utente può accedere a queste funzionalità selezionando il nodo contenente le immagini allineate e navigando nel menù *Modifica* → *Operazioni Stack*. Il sistema espone tre trategie classiche:

- **Somma (Sum):** Esegue l'addizione algebrica pixel per pixel, ideale per massimizzare il segnale delle debolissime strutture della chioma esterna.
- **Media (Mean):** Calcola la media aritmetica, ottimizzando l'abbattimento del rumore gaussiano e del rumore termico di fondo.
- **Mediana (Median):** Estrae il valore mediano per ogni coordinata spaziale. Questa opzione è di fondamentale importanza nell'astrofotografia cometaria: essendo la cometa in movimento rispetto al fondo cielo, le stelle (così come eventuali tracce satellitari o raggi cosmici) appaiono come artefatti transitori che si spostano da un fotogramma all'altro. Il filtro mediano scarta automaticamente queste anomalie, restituendo un'immagine della cometa pulita e isolata dalle interferenze.

Dal punto di vista dell'interazione utente, a differenza delle complesse fasi di allineamento, le operazioni di *stacking* sono state concepite per la massima reattività operativa. Per questa ragione, non vi è alcuna finestra di dialogo o parametrizzazione intermedia: il clic sulla voce di menù innesca immediatamente l'esecuzione del calcolo matematico in *background*.

Al termine del processo, in piena coerenza con il paradigma non distruttivo del software, l'immagine risultante non sovrascrive i dati di partenza, ma viene istanziata all'interno di un nuovo nodo figlio, pronto per essere sottoposto alle successive elaborazioni.

4.7.5 Ottimizzazione Spaziale: Ritaglio dei Fotogrammi (Cropping)

A chiusura della *pipeline* geometrica e di registrazione spaziale, Kometra [3] integra un modulo dedicato all'ottimizzazione dell'area di lavoro: lo strumento di Ritaglio (*Cropping*). Accessibile navigando nel menù principale alla voce *Modifica* → *Ritaglia*, questa funzionalità riveste un'importanza strategica fondamentale per l'efficienza dell'intero software.

Infatti, le moderne camere astronomiche producono matrici da decine di megapixel, ma la chioma cometaria e le sue delicate strutture interne occupano solo una ristretta porzione del campo inquadrato. Per questo motivo, ritagliare le immagini isolando esclusivamente il bersaglio scientifico permette di abbattere drasticamente la mole di pixel caricati in memoria. Di conseguenza, le successive e onerose operazioni di filtraggio morfologico (come le estrazioni tensoriali o i modelli radiali) risulteranno ordini di grandezza più veloci e reattive.

Dal punto di vista dell'ergonomia e della *User Experience*, la finestra mantiene la coerente struttura a due colonne adottata dal software. Mentre la porzione di destra è dedicata alla *viewport* visiva, la colonna di sinistra ospita le opzioni di configurazione. Qui l'operatore può specificare le dimensioni del riquadro di ritaglio (altezza e larghezza in pixel) e selezionare la logica di applicazione sull'intero blocco di immagini.

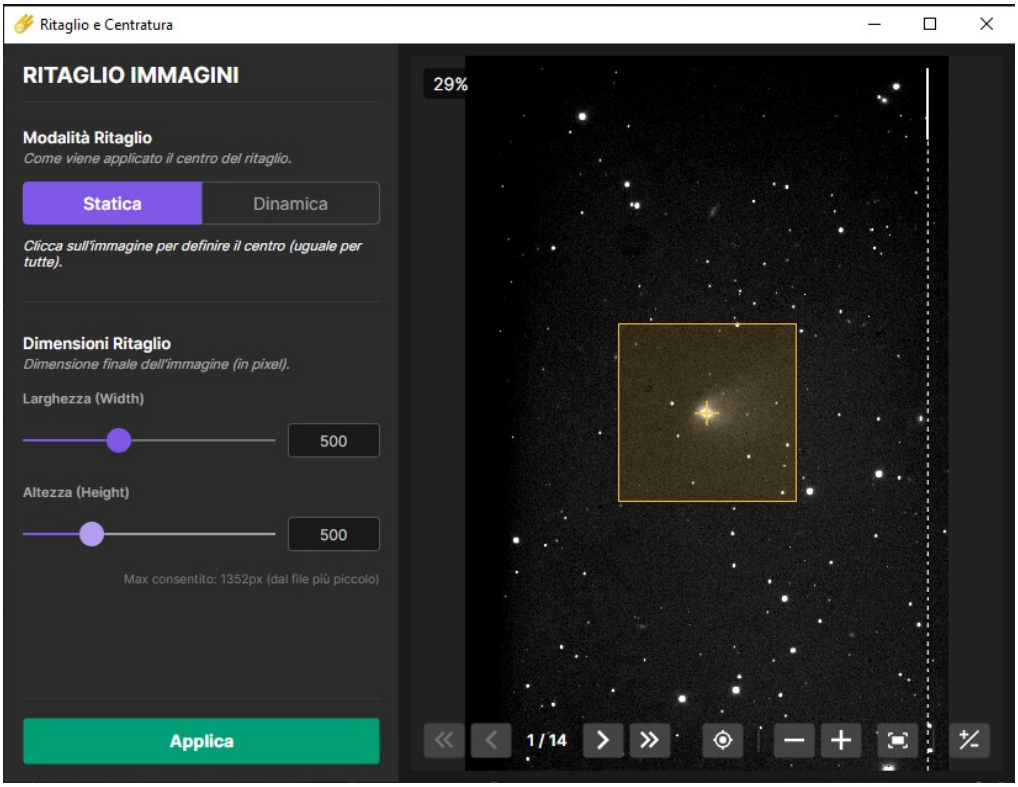

Figura 4.12. Interfaccia di Ritaglio (Cropping). A destra, il pannello di configurazione permette di impostare le dimensioni del *bounding box* e di selezionare la modalità operativa (Statica o Dinamica) per l'applicazione massiva all'intera sequenza temporale.

Operando frequentemente su nodi multipli (serie temporali), il sistema offre due metodologie di ritaglio distinte, studiate per adattarsi allo stato di calibrazione del *dataset*:

- **Modalità Statica:** Pensata per le sequenze di immagini che hanno già superato la fase di allineamento cometario. Poiché il nucleo della cometa è bloccato alle medesime coordinate in tutti i fotogrammi, le dimensioni e la posizione spaziale (X, Y) del riquadro di ritaglio vengono impostate una sola volta sull'immagine di riferimento e applicate in modo identico e simultaneo all'intero nodo.
- **Modalità Dinamica:** Concepita per sequenze grezze o non allineate. In questo scenario, le dimensioni del riquadro di taglio restano costanti per garantire coerenza dimensionale, ma l'operatore è chiamato a impostare manualmente la posizione del centro di ritaglio fotogramma per fotogramma, inseguendo visivamente lo spostamento del soggetto.

Capitolo 5

Sottrazione del Campo Stellare

5.1 Elaborazione starless

Separare gli oggetti celesti (Deep Sky Object (DSO)) dalle stelle di fondo è uno dei problemi più difficili nell'analisi astrofotografica. Se il bersaglio da analizzare è una cometa, il compito si complica ulteriormente a causa della sua natura dinamica e in continuo movimento.

Come analizzato nei capitoli precedenti, per compensare il moto proprio della cometa è necessario traslare i fotogrammi bloccando l'optocentro. Questa operazione fa sì che le stelle di sfondo si spostino continuamente da uno scatto all'altro. Se si procedesse all'integrazione (*stacking*) di questi fotogrammi senza alcun trattamento preventivo, le stelle genererebbero delle lunghe "strisce" luminose (*star trails*) che andrebbero a sovrapporsi, inquinare e spesso nascondere completamente i dettagli a bassissimo contrasto della coda cometaria e delle emissioni gassose.

La rimozione delle stelle (processo noto come *Starless*) si propone di risolvere questo problema alla radice, eliminando le sorgenti puntiformi da ogni singolo fotogramma prima della fase di stacking. I vantaggi di questa operazione sono molteplici:

- **Isolamento del segnale:** Permette di estrarre esclusivamente il flusso fotometrico appartenente alla cometa, rendendo possibili analisi morfologiche precise su getti e chioma. Infatti, spesso le stelle che si sovrappongono alla cometa causano problemi nell'identificazione corretta del nucleo.
- **Elaborazione non lineare indipendente:** Consente di adattare l'istogramma dell'immagine alla sola cometa per far emergere i dettagli deboli, senza saturare o gonfiare i profili stellari.
- **Estetica e pulizia del dato:** Evita la creazione di artefatti visivi durante lo stacking centrato sulla cometa.

5.1.1 Le Insidie del Processo: Artefatti e Integrità dei Dati

Nonostante i vantaggi, la rimozione stellare è un'operazione matematicamente distruttiva. Rimuovere i pixel contenenti una stella crea di fatto un "buco" (Not-a-Number (NaN) o zero) nella matrice dell'immagine. Il riempimento di questi vuoti, operazione nota in computer vision come *Inpainting*, richiede estrema cautela.

Le tecniche tradizionali di inpainting, spesso basate su semplici medie locali o sull'equazione di Laplace, tendono a generare due tipologie di artefatti disastrosi per l'astronomia:

1. **Artefatti di Ringing o Smudging:** Il buco viene riempito con un colore piatto e sfocato che interrompe bruscamente la granulometria naturale (*texture*) del fondo cielo, creando macchie visivamente artificiali.

2. **Rimozione accidentale del target:** Un'inadeguata parametrizzazione dell'algoritmo di segmentazione può indurre l'errata interpretazione delle condensazioni luminose della chioma come artefatti o aggregati stellari, portando alla perdita irreversibile di dati fotometrici cruciali.

Per far fronte a queste criticità, il software Kometra [3] implementa una pipeline di *Star Masking* e *Inpainting* proprietaria. A differenza dei software generalisti, questa architettura si affida a un approccio analitico e stocastico. Il processo è diviso in due macro-fasi: una rigida segmentazione morfologica per proteggere la cometa e isolare le stelle, seguita da un motore di ricostruzione stocastica che inietta rumore gaussiano calcolato localmente per preservare la fedeltà fotometrica del fondo cielo.

5.2 Segmentazione Morfologica: Costruzione delle Maschere

Prima di poter operare qualsiasi riempimento di pixel, è fondamentale stabilire una mappatura binaria (una maschera in cui il valore 255 indica i pixel bersaglio e lo 0 indica i pixel da ignorare). Questo processo, eseguito dalla classe `SegmentationEngine` tramite la libreria OpenCV, è rigorosamente scisso in due fasi sequenziali: la creazione di uno "scudo" per la cometa e la successiva estrazione delle stelle. Questa priorità è fondamentale per evitare la potenziale e irreversibile cancellazione delle strutture cometarie.

5.2.1 Isolamento del Bersaglio: La Maschera Cometaria

La prima operazione dell'algoritmo consiste nel localizzare la cometa per proteggerla in modo mirato dall'operazione di rimozione stellare. Avendo eseguito precedentemente l'allineamento (illustrato nel Capitolo precedente), l'algoritmo assume con assoluta certezza che l'optocentro della cometa si trovi nel pixel esatto $(W/2, H/2)$ dell'immagine.

Per determinare il metodo più robusto per isolare il corpo cometario, sono stati condotti diversi test esplorativi in ambiente Python:

- **Sogliatura Relativa Semplice:** Un primo test generava una maschera basata su una frazione percentuale del valore massimo dell'immagine. Pur essendo computazionalmente immediato, il metodo si è rivelato inaffidabile: la presenza di stelle saturate o difetti nel campo stellare alterava il valore massimo globale, sballando completamente la maschera.
- **Crescita di Regioni (*Region Growing*):** Un secondo approccio simulava l'espansione iterativa di una maschera a partire da un "seme" centrale, aggregando i pixel contigui con un'intensità affine a quella del nucleo. Pur tracciando accuratamente il gradiente della chioma, il costo computazionale delle iterazioni lo rendeva estremamente lento.
- **Segmentazione Dinamica Iterativa:** Un ultimo test impiegava algoritmi di rilevamento sorgenti applicando valori di soglia decrescenti in modo ciclico, supportati da una logica di ricerca spaziale in prossimità del fotocentro. Anche in questo caso, la necessità di reiterare il processo di segmentazione causava un collo di bottiglia prestazionale.

L'analisi di questi prototipi ha evidenziato un requisito cruciale per l'applicativo finale: la necessità di garantire un'interazione utentefluida e reattiva. Gli approcci puramente iterativi avrebbero infatti imposto prolungati tempi di attesa a ogni variazione dei parametri. Si è pertanto optato per una semplificazione della logica matematica, demandando gli oneri computazionali alla libreria ad alte prestazioni OpenCV e implementando un'architettura software asincrona. Tale soluzione consente all'operatore di regolare i parametri tramite interfaccia grafica e di visualizzare in tempo reale l'aggiornamento della maschera in sovrimpressione (*overlay*), ottenendo un riscontro visivo istantaneo.

La pipeline operativa definitiva si articola come segue:

Inizialmente, l'immagine viene sottoposta a una sottrazione globale del fondo cielo, stimato su base statistica (*background level*), al fine di abbattere il rumore termico e di lettura isolando le componenti di emissione attive. Successivamente, il software esegue una singola binarizzazione (*Thresholding*): la soglia di taglio non viene determinata per via euristica o per tentativi, bensì è definita analiticamente dall'utente (`CometThresholdSigma`) e moltiplicata per la deviazione standard del rumore (σ). In questo modo, il taglio si adatta in un solo passaggio alla naturale rumorosità dello scatto.

Poiché la binarizzazione estrae sia la cometa che le stelle brillanti, il software impiega l'algoritmo di *Connected Components*, che assegna un'etichetta numerica a ciascun gruppo di pixel contigui. Ispezionando il pixel centrale, l'intera isola a esso collegata viene confermata come "Corpo Cometario". Qualora il centro esatto fosse vuoto a causa di asimmetrie anomale o di un ritaglio sfalsato, entra in gioco la logica di *fallback* derivata dai test preliminari: il sistema analizza un'area circostante di raggio 5 pixel e aggancia l'etichetta maggiormente presente.

Una volta isolata la componente connessa, l'algoritmo applica una chiusura morfologica (*Morphological Close*) con un elemento strutturante ellittico 5x5 per saturare eventuali micro-fratture causate dal rumore. Se l'utente lo richiede tramite l'interfaccia, la maschera subisce una dilatazione iterativa con un kernel 3x3 per espanderne l'area di copertura. Infine, l'algoritmo esegue una ricerca dei contorni esterni (*FindContours*) e li riempie completamente di bianco. Questo garantisce la creazione di uno "scudo" di protezione solido e ininterrotto per il target.

5.2.2 La Mappatura del Campo Stellare: Estrazione e Pulizia

Avendo creato lo scudo cometario, il software è ora pronto a mappare il campo stellare. Il processo prende nuovamente l'immagine originaria priva di fondo cielo, ma esegue un'operazione preliminare fondamentale: tutti i pixel che ricadono all'interno dell'area dello scudo cometario appena calcolato vengono forzati a zero. La cometa viene, di fatto, "spenta" prima della ricerca delle stelle, impedendo che i suoi gradienti luminosi vengano erroneamente interpretati come target stellari.

La rilevazione accurata del campo stellare ha richiesto un'intensa fase di prototipazione in Python, durante la quale sono stati testati algoritmi fotometrici avanzati:

- **Astrometria Classica:** Sono stati impiegati algoritmi di identificazione puntuale che valutano il profilo gaussiano (Full Width at Half Maximum (FWHM)) delle sorgenti. Per includere correttamente gli aloni, il sistema calcolava una soglia adattiva basata sulla mediana locale di una "patch" ritagliata attorno a ciascun centroide rilevato.
- **Laplaciano del Gaussiano (*Blob Laplacian of Gaussian (LoG)*):** È stato testato un approccio multi-scala per il rilevamento di "blob" luminosi, calcolando l'intersezione di filtri gaussiani a diverse deviazioni standard.
- **Segmentazione e Analisi di Forma:** Sono stati provati approcci basati sull'estrazione dei contorni (*Canny Edge Detection*) e sull'analisi dell'elongazione per isolare stelle mosse o tracce (*trails*), con logiche di ritentativo iterativo nel caso in cui le aree segmentate risultassero eccessivamente ampie.

Sebbene i test preliminari garantissero un'estrema precisione fotometrica, l'analisi prestazionale ha evidenziato un limite computazionale insostenibile. Il calcolo di metriche locali e profili gaussiani per decine di migliaia di stelle imponeva tempistiche incompatibili con l'interattività richiesta dal software. Ai fini della mascheratura, infatti, l'obiettivo non è la misurazione rigorosa del flusso stellare, bensì la rapida ed efficiente generazione di una copertura spaziale.

Nella versione definitiva del software, la logica è stata drasticamente snellita e parallelizzata utilizzando operazioni matriciali globali tramite OpenCV. Dopo aver oscurato la cometa, viene applicato un taglio di soglia globale basato su un parametro fornito dall'utente moltiplicato per la deviazione standard del rumore. Si genera così una mappa binaria grezza che contiene stelle reali, ma inevitabilmente anche artefatti del sensore (come gli *hot pixels*) e rumore impulsivo, tipicamente costituiti da aggregati di pochissimi pixel.

Per discriminare le stelle reali dal rumore elettronico ad alta frequenza senza ricorrere a lenti algoritmi di *source detection*, il sistema utilizza un approccio puramente morfologico basato sul parametro `MinStarDiameter`. Se questo valore supera 1, l'algoritmo crea un elemento strutturante ellittico di raggio corrispondente e applica un'operazione di Apertura (*Morphological Open*), che consiste in un'erosione seguita da una dilatazione. Questa operazione matematica "lima" via automaticamente tutte le macchie luminose il cui diametro è inferiore a quello della sonda strutturale passante, polverizzando di fatto il rumore termico in una singola passata matriciale senza intaccare la posizione delle stelle reali.

L'ultimo e decisivo passaggio è la Dilatazione Stellare (`StarDilation`). Una stella in un'immagine astronomica non termina mai dove la binarizzazione l'ha tagliata; possiede un debole alone luminoso (*glow*) indotto dallo scattering atmosferico e ottico che sfuma dolcemente nel fondo cielo. Se si omettesse di dilatare la maschera delle stelle per includere questi aloni invisibili, il successivo algoritmo di *inpainting* (riempimento) non avrebbe dati sufficienti e genererebbe un artefatto scuro (*dark ringing*) attorno all'area interpolata. La maschera pulita viene quindi iterativamente dilatata con un kernel 3x3 per espandere i dischi stellari, includendo le loro propaggini fotometriche e rendendoli pronti all'eliminazione finale.

5.3 Ricostruzione del Segnale: Inpainting Stocastico

L'operazione di mascheramento, descritta nella sezione precedente, elimina fisicamente le stelle dall'immagine sostituendole con dei "vuoti" (NaN o zeri). Il processo matematico che si occupa di riempire questi buchi, rendendo la modifica invisibile all'occhio umano e ininfluente per gli algoritmi statistici di stacking, prende il nome di *Inpainting*.

Poiché le immagini astronomiche del cielo non possiedono una tinta unita, ma sono caratterizzate da un rumore di fondo naturale continuo e fondamentale per preservare la linearità dei dati, la scelta dell'algoritmo di ricostruzione è risultata essere la fase più delicata dell'intero sviluppo.

Per individuare la soluzione matematicamente più corretta, la fase iniziale di Ricerca e Sviluppo (Research and Development (R&D)) ha previsto lo sviluppo di prototipi in Python volti a esplorare tre filosofie di ricostruzione concettualmente molto diverse tra loro:

- **Interpolazione Pura (Approccio Geometrico-Spaziale):** Il primo modello tentava di risolvere i buchi trattando l'immagine come una superficie matematica continua. L'algoritmo analizzava i pixel sui bordi della maschera e decideva il valore dei pixel intermedi unendo gli estremi. Sebbene spazialmente corretta, questa logica si è rivelata inaccettabile per l'astrofotografia: l'interpolazione lineare genera campiture perfettamente lisce che, su un fondo cielo dominato dal rumore termico e di Poisson, spiccano come evidenti macchie artificiali. Aggiungere a posteriori un rumore globale per mascherare il difetto non risolveva il problema, poiché ignorava le fisiologiche variazioni locali della deviazione standard del fondo.
- **Stima del Background 2D (Approccio Fotometrico a Bassa Frequenza):** Il secondo tentativo ha cambiato radicalmente prospettiva, passando dalla pura geometria alla fotometria. Utilizzando la libreria `photutils`, si è calcolato un modello bidimensionale delle variazioni a grande scala (*macro-blocchi*) del fondo cielo. L'idea concettuale era ripristinare il corretto gradiente luminoso sottostante, sostituendo le stelle con il livello di segnale "puro" calcolato in quella zona. Tuttavia, pur rispettando i lenti chiaroscuri del cielo, questo metodo restituiva valori numerici piatti, distruggendo del tutto l'indispensabile continuità granulare (*texture*) del rumore adiacente.
- **Campionamento Casuale Uniforme Locale (Approccio Stocastico ad Alta Frequenza):** I fallimenti precedenti hanno dimostrato che il rumore locale non è un "difetto" da ignorare, ma la componente visiva fondamentale da ricostruire. Il terzo prototipo ha quindi abbandonato le superfici e i gradienti continui in favore della sintesi di *texture*. Analizzando una piccola finestra di pixel "sani" attorno al buco, il sistema ne estraeva i valori estremi (minimo e massimo) e calcolava il pixel mancante pescandolo da una distribuzione

di probabilità uniforme. Questo approccio ripristinava finalmente la "rumorosità", mimetizzando perfettamente i buchi all'occhio umano. Il suo limite risiedeva unicamente nella validità fisica della statistica: una distribuzione uniforme genera valori con la stessa probabilità, un comportamento che non rispecchia la vera fisica dei sensori ottici digitali, il cui rumore è notoriamente distribuito secondo una curva a campana (distribuzione Normale o Gaussiana).

L'analisi di questi prototipi ha delineato i tre requisiti fondamentali che l'algoritmo finale in C# avrebbe dovuto soddisfare rigorosamente:

1. Adattarsi dinamicamente all'intensità luminosa *locale* e non globale.
2. Generare valori utilizzando una distribuzione Gaussiana (Normale) e non uniforme.
3. Mantenere un'elefate efficienza computazionale, essenziale per processare immagini da svariati megapixel in frazioni di secondo.

Per soddisfare questi stringenti requisiti nella release finale del software, è stato implementato un motore di *inpainting* (`InpaintingEngine`) interamente dedicato, il quale coniuga tecniche di campionamento Monte Carlo a ottimizzazioni di memoria a basso livello.

Inizialmente, il sistema impiega le librerie OpenCV per combinare la maschera delle sorgenti stellari rilevate con i pixel nulli (NaN). Al fine di assicurare che il campionamento finalizzato alla ricostruzione estragga esclusivamente il reale segnale del fondo cielo, l'algoritmo calcola maschera di sicurezza (*Safe Mask*) dilatando ulteriromente i contorni stellari. In questo modo, le code periferiche degli aloni stellari non inquinano le statistiche locali del *background*.

L'accesso ricorsivo ai singoli pixel tramite librerie di elaborazione immagini (es. `Mat.At` di OpenCV) introduce un massiccio ritardo (*overhead*). Per massimizzare le prestazioni, l'intera matrice 2D dell'immagine e le relative maschere vengono "appiattite" e riversate in vettori monodimensionali (array 1D) adiacenti nella memoria gestita (tramite `Marshal.Copy`). Questo approccio consente ai core della CPU di operare direttamente sui registri della memoria RAM in modo contiguo, annullando i tempi di latenza.

La ricostruzione avviene in parallelo (`Parallel.ForEach`) su tutti i pixel mancanti. Invece di ispezionare tutti i pixel all'interno di una finestra NxN (operazione che comporterebbe un costo computazionale quadratico), l'algoritmo adotta un metodo basato su campionamenti casuali: per ciascun pixel vuoto, esplora una finestra locale (inizialmente di raggio 7 pixel, espandibile dinamicamente fino a un massimo di 75) lanciando coordinate casuali per sondare i pixel adiacenti. L'obiettivo è raccogliere rapidamente almeno 8 (fino a 15) valori estratti dalla *Safe Mask*, senza dover leggere l'intera finestra. Su questi campioni sani, il sistema calcola la media (μ) e la varianza statistica (σ^2) del fondo locale.

Definiti μ e σ locali, il sistema genera il pixel sintetico utilizzando l'algoritmo di Box-Muller, che trasforma due variabili casuali a distribuzione uniforme in una variabile a distribuzione normale $\mathcal{N}(\mu, \sigma^2)$. Per impedire che i thread collidano durante la generazione dei numeri casuali (il problema del *thread-locking* tipico della classe `System.Random`), è stato implementato un generatore pseudo-casuale lineare inline ad altissime prestazioni (*Xorshift*). Il *seed* iniziale per ogni pixel è calcolato univocamente a partire dalle sue coordinate spaziali (x, y) e dal ciclo di iterazione, garantendo la totale indipendenza dei thread (*Zero-Lock concurrency*) e una riproducibilità matematica esatta.

Il risultato di questa complessa operazione invisibile è una maschera stellare completamente ricostruita in breve tempo: i "buchi" vengono saturati con un rumore di Poisson simulato che copia esattamente le variazioni luminose e la deviazione standard del fondo circostante, rispettando a pieno l'integrità fotometrica della ripresa astronomica. È tuttavia doveroso specificare che il risultato dell'inpainting su un singolo fotogramma potrebbe a volte presentare lievi imperfezioni visive, dovute alla natura caotica e imprevedibile del rumore locale. La reale potenza di questa tecnica si manifesta infatti solo al termine del processo elaborativo globale, ovvero dopo aver effettuato l'integrazione (*stacking*) di fotogrammi multipli. Poiché l'allineamento viene effettuato

seguendo il moto della cometa, le stelle di sfondo e le relative aree ricostruite subiscono uno spostamento continuo da uno scatto all'altro. La successiva somma statistica dei pixel fa sì che queste minime imprecisioni si comportino come rumore scorrelato, venendo mediate e dissolvendosi quasi completamente nel fondo cielo. Come evidenziato dal confronto visivo riportato nella Figura 5.1, l'integrazione di immagini preventivamente trattate con questa architettura restituisce un risultato nettamente superiore rispetto allo stacking tradizionale: i difetti si annullano, le strisce luminose si estinguono e il segnale cometario puro viene isolato con successo.

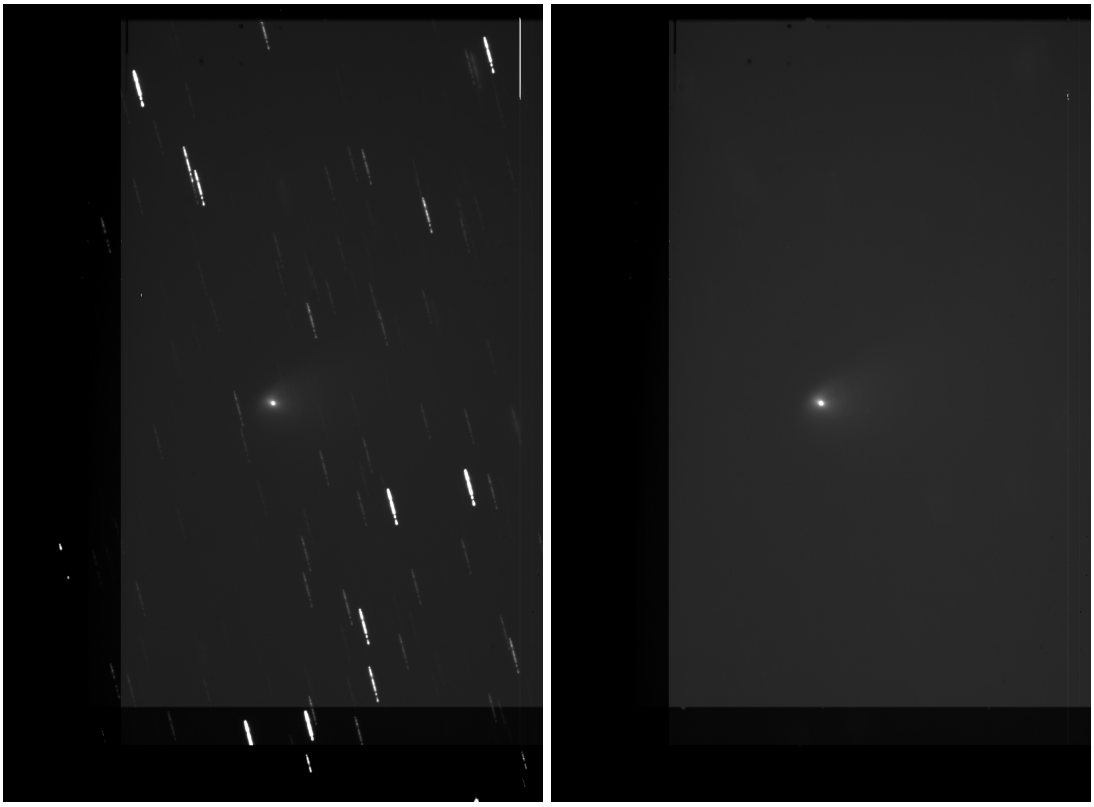

Figura 5.1. Confronto visivo dei risultati di *stacking* con allineamento sulla cometa. A sinistra, l'integrazione tradizionale in cui il moto siderale genera evidenti e fastidiose strisce luminose (*star trails*). A destra, il risultato dell'integrazione effettuata dopo l'applicazione della pipeline di rimozione stellare: le minime imperfezioni di *inpainting* sui singoli fotogrammi si mediano statisticamente scomparendo nel fondo cielo, lasciando il target perfettamente isolato.

5.4 Architettura Software: Segmentazione e Inpainting

Per garantire manutenibilità, testabilità e alte prestazioni, l'implementazione in C# della rimozione stellare non è stata concepita come uno script monolitico, ma ingegnerizzata seguendo i rigidi paradigmi della programmazione orientata agli oggetti e della *Dependency Injection*. Il sistema si basa su una netta separazione delle competenze, demandando il flusso di lavoro a un coordinatore centrale e i calcoli matematici a motori specializzati.

Per mantenere la logica di dominio perfettamente isolata, il controllo delle operazioni è affidato alla classe `MaskingCoordinator`. Questo modulo riceve i file di input nel formato FITS e i parametri operativi impostati dall'utente, comportandosi come un vero e proprio direttore d'orchestra. Il coordinatore nasconde la complessità dell'intera *pipeline* matematica: richiede i dati grezzi al manager dei file per convertirli in matrici operative e ne esegue il flusso sequenziale in modo asincrono. Nello specifico, richiama il calcolo delle statistiche, avvia la generazione delle maschere e infine delega l'applicazione dell'algoritmo di inpainting. Inoltre, si occupa di gestire l'elaborazione iterativa massiva su intere cartelle di file (*Batch Processing*), fornendo report di

avanzamento progressivi e aggiornando i metadati degli header FITS con lo storico delle operazioni eseguite, il tutto senza bloccare il thread principale dell'applicazione.

Al livello più profondo dell'architettura operano invece i motori di calcolo, ovvero classi progettate per essere prive di stato e focalizzate esclusivamente sulle prestazioni computazionali, nel pieno rispetto del principio di singola responsabilità. Queste componenti vengono istanziate e iniettate nel coordinatore tramite interfacce e si dividono in tre competenze matematiche specifiche.

Il primo a intervenire è il `RadiometryEngine`, il modulo deputato all'analisi statistica dell'immagine. Esso calcola metriche robuste come la mediana e la deviazione standard del rumore (σ) tramite logiche di *sigma-clipping*, fornendo i valori di base essenziali per tutte le sogliature successive.

Nella fase successiva interviene il `SegmentationEngine`, modulo che incapsula la logica di *computer vision* avvalendosi dell'implementazione C# della libreria OpenCV. Questa classe espone metodi dedicati e indipendenti per l'analisi della cometa e per l'individuazione del campo stellare, gestendo in modo efficiente la binarizzazione, l'estrazione delle componenti connesse e l'applicazione delle operazioni morfologiche (quali erosione, dilatazione e chiusura) necessarie per isolare spazialmente i bersagli di interesse.

Infine, il core matematico del riempimento stocastico è racchiuso nell'`InpaintingEngine`. Ricevendo la matrice dell'immagine bucata e la maschera stellare definitiva, questo motore si occupa di "appiattire" la memoria bidimensionale in array monodimensionali contigui per massimizzare la velocità di accesso alla RAM. Subito dopo, esegue l'algoritmo di campionamento e la generazione del rumore gaussiano sfruttando il calcolo parallelo multi-core in totale assenza di blocchi fra i thread (*zero-lock concurrency*).

Grazie a questa rigida modularità, il codice finale risulta non solo estremamente reattivo e veloce, ma anche facilmente estensibile: eventuali aggiornamenti futuri agli algoritmi morfologici o di ricostruzione del segnale richiederanno unicamente la sostituzione del rispettivo motore matematico, lasciando intatta l'intera infrastruttura di coordinamento.

5.5 Interfaccia Utente: Mascheramento e Interazione Operativa

Infine, questo capitolo si conclude con la descrizione dell'interfaccia grafica progettata per questa funzionalità, con lo scopo di rendere il complesso processo di mascheramento e rimozione stellare non solo intuitivo, ma altamente interattivo. L'accesso a questa specifica funzionalità è governato in modo contestuale: la relativa voce di menu, posizionata sotto il percorso *Modifica* → *Maschera Stelle*, risulta inizialmente disabilitata. Essa diviene selezionabile esclusivamente dopo che l'utente ha selezionato un nodo valido all'interno della Board di lavoro, garantendo così che lo strumento di elaborazione venga invocato solo in presenza di un set di dati caricato, coerente e pronto per essere processato.

Una volta attivata, la finestra operativa si presenta strutturata in due sezioni principali, studiate per ottimizzare il flusso di lavoro visivo (come illustrato nella Figura 5.2). Sulla porzione sinistra dello schermo sono raggruppati tutti i controlli e i parametri matematici modificabili dall'operatore, mentre l'intera area destra è dedicata a un'ampia viewport grafica. Quest'ultima mostra l'immagine attualmente selezionata all'interno del nodo e, sfruttando i motori di calcolo asincroni descritti nei capitoli precedenti, proietta in tempo reale le maschere colorate sovrapposte all'immagine man mano che i valori vengono alterati. Per facilitare l'immediata distinzione visiva dei bersagli sul fondo cielo, la maschera di protezione della cometa viene tracciata in giallo, mentre quella dedicata alla rimozione delle stelle viene resa in blu. L'interfaccia offre inoltre la massima flessibilità consentendo di attivare o disattivare la visualizzazione di ciascuna maschera in modo del tutto indipendente; in questo modo l'operatore può scegliere se mantenerle entrambe attive per valutarne l'intersezione complessiva, oppure isolarne una singola per un'ispezione più minuziosa dei dettagli morfologici.

Per quanto riguarda il controllo logico, i parametri a disposizione dell'utente sono stati volutamente essenzializzati per evitare inutili complessità. Per il mascheramento del bersaglio principale, ovvero il nucleo e la chioma della cometa, è possibile regolare unicamente il valore di soglia statistica e la quantità di dilatazione morfologica espressa in pixel, permettendo all'utente di gonfiare l'area di protezione visiva fino a coprire in totale sicurezza l'intero target. Allo stesso modo, per la definizione della maschera stellare vengono riproposti i cursori di soglia e dilatazione per calibrare l'espansione sugli aloni luminosi, a cui si aggiunge un parametro critico aggiuntivo rappresentato dal diametro minimo. Quest'ultimo filtro spaziale permette al software di discriminare efficacemente le stelle reali dal rumore termico ad alta frequenza, istruendo l'algoritmo a ignorare automaticamente tutte le porzioni luminose che non raggiungono la dimensione fisica attesa per un vero disco stellare.

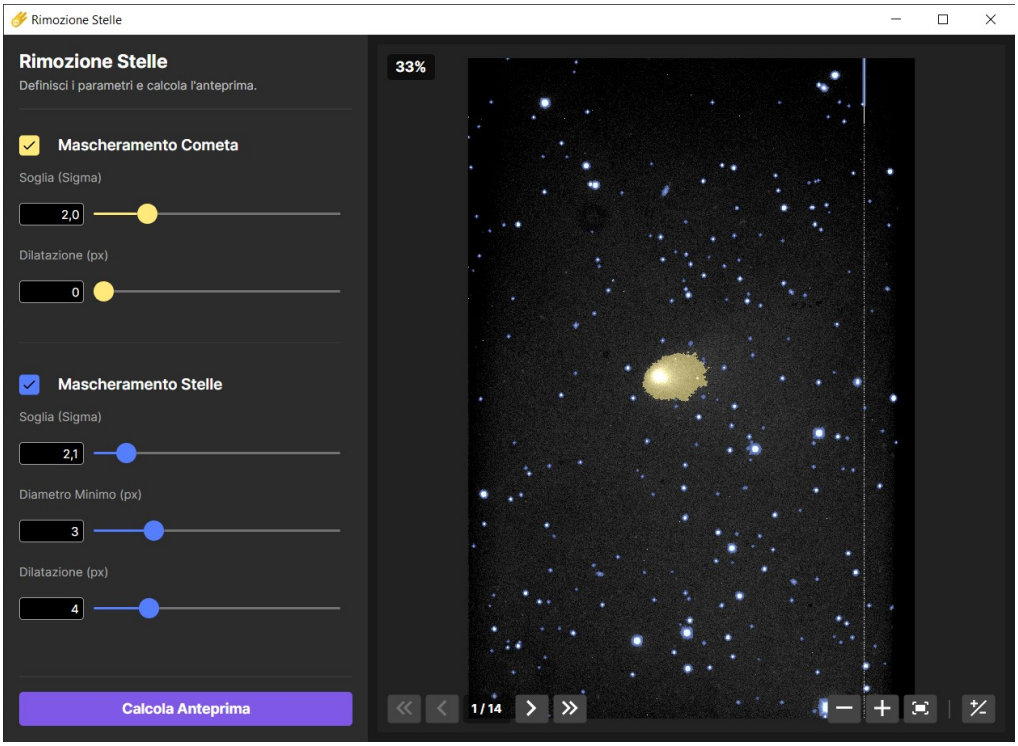

Figura 5.2. Interfaccia di controllo per il mascheramento: a sinistra i parametri di configurazione morfologica, a destra l'anteprima dinamica delle maschere generate (cometa in giallo, stelle in blu) in sovrapposizione al campo inquadrato.

Al termine della fase di taratura visiva delle maschere, il flusso di lavoro prevede una verifica formale sul campo tramite un apposito pulsante per il calcolo dell'anteprima. Cliccando su tale comando, il software esegue l'operazione completa di rimozione e inpainting stocastico, applicandola in pochi istanti esclusivamente all'immagine correntemente visualizzata nella viewport. Questo fondamentale passaggio intermedio permette all'utente di ispezionare la qualità della ricostruzione del fondo cielo prima di impegnare le risorse di sistema. Se il risultato estetico e fotometrico risulta soddisfacente, è possibile confermare l'operazione: il coordinatore prenderà in carico la trasformazione applicandola in modalità batch a tutte le immagini contenute nel nodo selezionato. In caso contrario, l'operazione può essere annullata per consentire all'operatore di tornare ai parametri e procedere a un'ulteriore raffinazione.

Capitolo 6

Filtri morfologici

Come delineato nei capitoli precedenti, l'elaborazione di immagini cometarie impone il superamento di sfide fisiche e computazionali profondamente diverse da quelle riscontrate nell'astrofotografia convenzionale. Dopo aver analizzato le criticità legate all'acquisizione del dato grezzo e aver descritto i meccanismi di rilevamento e tracciamento ad altissima precisione del centroide nucleare, il presente capitolo si focalizza sul cuore analitico del software: il motore di filtraggio morfologico.

6.1 Obiettivi del Filtraggio Morfologico

Risolto il problema dell'allineamento geometrico e del mascheramento del campo stellare, l'ostacolo principale all'estrazione di informazioni scientificamente rilevanti risiede nel gradiente di luminosità della chioma cometaria. L'intensità luminosa emessa dai gas e dalle polveri decade in modo estremamente repentino allontanandosi dal nucleo, seguendo tipicamente un profilo inversamente proporzionale alla distanza radiale spaziale ($1/\rho$). Questo gradiente luminoso, dominato dalla componente simmetrica dell'emissione, agisce di fatto come un forte velo fotometrico. Esso possiede una gamma dinamica talmente estesa da sommergere e nascondere alla vista qualsiasi struttura morfologica a basso contrasto, rendendo inefficaci le tradizionali operazioni lineari o le semplici curve di *stretching* dell'istogramma.

L'obiettivo fondamentale dell'applicazione dei filtri spaziali sviluppati all'interno di Kometra [3] non è pertanto di natura estetica, bensì strettamente analitica: isolare le deboli anomalie strutturali dal gradiente predominante della chioma intatta. Tramite l'impiego di operatori matematici e trasformazioni geometriche, gli algoritmi implementati mirano a sopprimere in modo selettivo la componente isotropa e simmetrica dell'emissione cometaria, permettendo di far emergere con maggior chiarezza le anisotropie fisiche dell'oggetto, come getti di polvere emessi radialmente da sorgenti discrete, ma che a causa della rotazione del nucleo e in base all'orientamento dell'asse di rotazione possono assumere varie forme: lineari, a ventaglio più o meno ampio, o a spirale che crea "gusci" concentrici in espansione (shells).

Per permettere un'indagine così approfondita, si è scelto di adottare un approccio fortemente multidisciplinare, integrando algoritmi provenienti da domini scientifici eterogenei. Da un lato vengono impiegate metodologie astronomiche standard, essenziali per la modellazione della chioma, come i filtri basati sui gradienti radiali e rotazionali (es. Larson-Sekanina) o la sottrazione di modelli sintetici. Dall'altro, il software attinge a piene mani dalla moderna *computer vision*, integrando tecniche di normalizzazione statistica locale e di equalizzazione adattiva. Infine, a titolo fortemente sperimentale, sono state riadattate tecniche derivate dall'*imaging* biomedico: è il caso del filtro di Frangi, originariamente concepito per il tracciamento vascolare e qui impiegato per l'identificazione tensoriale dei getti cometari.

Tuttavia, l'applicazione di filtri non lineari e differenziali su matrici di dati astrofisici comporta il rischio intrinseco di introdurre artefatti matematici, ovvero di generare strutture fittizie derivanti unicamente da errori di discretizzazione o da un errato posizionamento del centroide di rotazione. Per tale motivo, lo sviluppo dei motori di elaborazione è stato guidato dalla necessità di fornire

all'utente una suite di filtri basati su principi matematici profondamente diversi tra loro. La possibilità di elaborare la medesima sequenza FITS parallelamente tramite l'architettura a nodi, confrontando i risultati di una sottrazione rotazionale, di una normalizzazione statistica locale o di un'analisi tensoriale, fornisce ai ricercatori il mezzo per validare scientificamente la reale natura fisica della struttura osservata, scartando eventuali artefatti computazionali.

Nelle sezioni seguenti verranno analizzati nel dettaglio i principali algoritmi di filtraggio morfologico implementati nell'applicativo, partendo dai fondamenti teorici e dalla formalizzazione matematica, fino a giungere alle specifiche scelte architetturali adottate per la loro traduzione in codice C# nativo ad alte prestazioni.

6.2 Modelli Radiali e Sottrazione Globale della Chioma

La morfologia della chioma di una cometa è, in prima approssimazione, dominata da un'espansione isotropa di gas e polveri. Questa espansione genera un profilo fotometrico fortemente simmetrico rispetto al centroide del nucleo. Il principio fondante dei filtri geometrici qui esposti risiede nell'ipotesi che le strutture di interesse scientifico costituiscano delle *anisotropie* sovrapposte a tale profilo simmetrico. Rimuovendo matematicamente la componente radiale globale, parametrizzata rispetto al centro della cometa, è possibile isolare il segnale ad alta frequenza spaziale.

6.2.1 Il Filtro Rotazionale di Larson-Sekanina

Il filtro rotazionale, originariamente proposto dagli astronomi Stephen Larson e Zdenek Sekanina nel 1984 per lo studio dei getti della Cometa di Halley [7], rappresenta l'algoritmo capostipite per l'analisi morfologica cometaria. Il principio di funzionamento è concettualmente semplice: l'immagine originale viene duplicata e ruotata di un piccolo angolo θ attorno al centroide del nucleo. Successivamente, l'immagine ruotata viene sottratta dall'immagine originale.

Nell'implementazione sviluppata all'interno del motore `GradientRadialEngine`, il software offre due varianti distinte del filtro:

- **Variante Asimmetrica:**
  Definendo l'intensità del pixel in coordinate polari con origine nel nucleo come $I(\rho, \phi)$, l'operazione di base prevede una singola rotazione di un angolo θ. L'equazione spaziale applicata è:
  $$I_{Asimmetrico}(\rho, \phi) = I(\rho, \phi) - I(\rho, \phi + \theta)$$
  Poiché la chioma simmetrica di fondo presenta variazioni trascurabili per piccole variazioni dell'angolo azimutale ϕ, la sottrazione cancella quasi interamente il gradiente radiale. Al contrario, una struttura lineare stretta come un getto, non sovrapponendosi perfettamente alla sua controparte ruotata, genera un pattern visivo derivativo, caratterizzato da un'alternanza di picchi positivi (il getto reale) e negativi (l'"ombra" sottratta). Nel codice, questo si traduce in una semplice chiamata `Cv2.Subtract`.

- **Variante Simmetrica:**
  Per bilanciare visivamente l'effetto derivativo ed evitare il micro-spostamento apparente del baricentro del getto, il software implementa una variante simmetrica. In questo caso, l'immagine subisce una doppia rotazione spaziale ($+\theta$ e $-\theta$) e i due risultati vengono prima sommati tra loro. Poiché la somma di due immagini ruotate produce un fondo luminoso di intensità doppia rispetto allo scatto di partenza, sottrarlo direttamente porterebbe a un severo troncamento dei dati verso i valori negativi (*clipping*). Per preservare il corretto bilanciamento fotometrico (*baseline*), l'algoritmo raddoppia matematicamente l'intensità dell'immagine originale prima della sottrazione:
  $$I_{Simmetrico}(\rho, \phi) = 2I(\rho, \phi) - [I(\rho, \phi + \theta) + I(\rho, \phi - \theta)]$$
  A livello di codice, questa ottimizzazione viene eseguita in un singolo passaggio sfruttando l'istruzione vettorializzata `Cv2.ScaleAdd(src, 2.0, -sumRot, dst)`, che scala l'immagine

sorgente di un fattore 2.0 garantendo un risultato centrato attorno allo zero termico. Un esempio dell'efficacia di questa elaborazione, applicata con un angolo di 45° su un'immagine preventivamente privata del campo stellare, è visibile nel pannello di confronto di Figura 6.1.

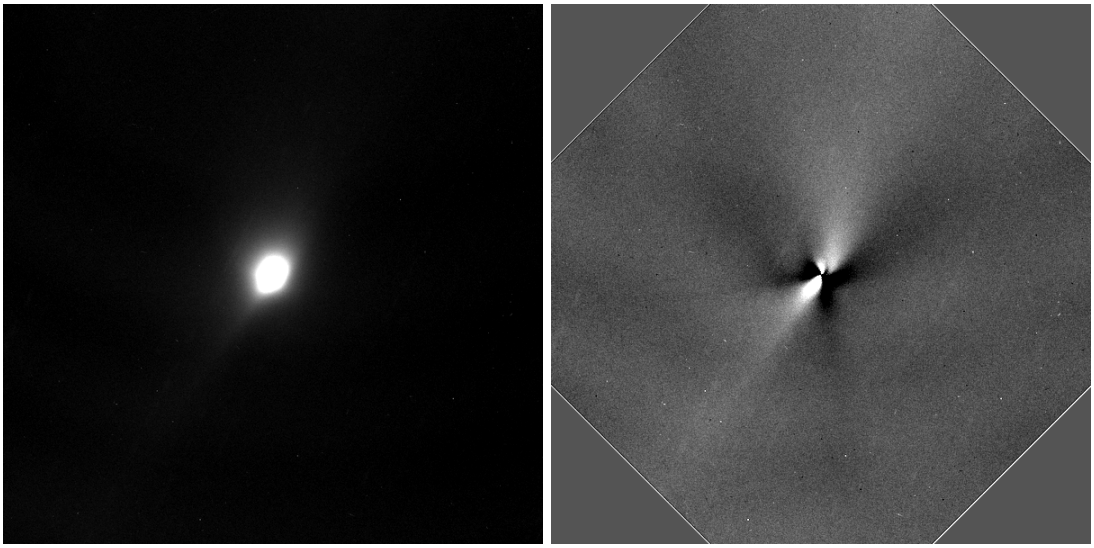

Figura 6.1. Applicazione del filtro rotazionale di Larson-Sekanina sulla cometa 67P. A sinistra, l'immagine originaria ripulita dal campo stellare evidenzia un gradiente radiale dominante. A destra, l'impiego del filtro con angolo di 45° neutralizza la componente diffusa della chioma, isolando e contrastando nettamente i getti interni.

Per quanto concerne l'architettura software, il calcolo non avviene tramite costose trasformazioni polari, bensì operando direttamente e in modo asincrono nel dominio cartesiano. Per garantire la massima fedeltà del dato, è cruciale che il centro di rotazione coincida con il centroide subpixel del nucleo. Il software sfrutta le routine ottimizzate di OpenCV (`Cv2.GetRotationMatrix2D`) per generare la matrice affine di trasformazione 2×3. A questa matrice viene successivamente applicata una manipolazione algebrica diretta sui termini di traslazione pura per iniettare i parametri di *radial shift* (s_x, s_y), permettendo all'utente di correggere dinamicamente le derive del baricentro.

Al fine di limitare la perdita di risoluzione intrinseca nelle trasformazioni geometriche discrete, la riproiezione spaziale dei pixel (*Warping*) viene eseguita adottando un algoritmo di interpolazione bicubica, il quale valuta il vicinato di 4×4 pixel minimizzando gli artefatti di *aliasing*.

Un problema critico insito nelle rotazioni matriciali riguarda la gestione dei bordi (operazioni *out-of-bounds*). Quando un'immagine viene ruotata, i vuoti generati ai vertici della matrice vengono tipicamente riempiti con zeri (costante di *padding*). Sottraendo tali regioni all'immagine di partenza, si genererebbero evidenti artefatti ad arco ai margini del campo visivo. Per neutralizzare questa anomalia, la funzione genera una *dummy mask* (una maschera logica composta interamente da pixel validi). Questa maschera subisce le medesime trasformazioni affini dell'immagine scientifica. Nel caso simmetrico, le maschere ruotate vengono intersecate tramite l'operatore logico `BitwiseAnd`, determinando l'esatta area di sovrapposizione valida. Infine, la maschera normalizzata viene moltiplicata per l'immagine risultante (`Cv2.Multiply`), azzerando qualsiasi pixel che non possieda una controparte reale in tutti i fotogrammi coinvolti nell'operazione.

6.2.2 Modello Mediano della Chioma (MCM)

A differenza delle tecniche basate su derivate spaziali, l'algoritmo MCM [47] adotta un approccio non invasivo per ricostruire un modello di chioma sintetica direttamente dall'immagine originale. L'obiettivo è sintetizzare una chioma perfettamente simmetrica da utilizzare come termine di confronto globale.

Per ottenere un risultato scientificamente valido, l'algoritmo richiede come prerequisito l'individuazione del fotocentro della cometa. Nel contesto di Kometra [3], essendo le immagini già state preventivamente allineate e centrate sul nucleo, il software assume il centro geometrico della matrice come origine per la mappatura in uno spazio polare $P(\rho, \theta)$. Da questa posizione, vengono estratti i valori dei pixel posti su serie di cerchi concentrici. Ogni cerchio viene trattato come un vettore di valori di cui viene calcolato il valore mediano [47]:

$$M(\rho) = \text{mediana}\{P(\rho, \theta_1), P(\rho, \theta_2), \ldots, P(\rho, \theta_n)\}$$

Utilizzando questi valori mediani, il software costruisce un modello di chioma sintetica composto da cerchi della medesima intensità. Questo modello può essere successivamente sottratto dall'immagine originale o utilizzato come divisore per evidenziare le strutture nascoste [47].

Una fase di pre-elaborazione fondamentale riguarda la sanitizzazione dei campioni: prima di procedere all'ordinamento, il software filtra i dati escludendo i valori nulli o non validi che potrebbero distorcere il modello (ad esempio a causa di pixel con intensità prossima allo zero o NaN):

$$P_{\text{valid}}(\rho) = \{v \in P(\rho, \theta) \mid |v| > 10^{-9} \wedge v \neq \text{NaN}\} \tag{6.1}$$

Per garantire un'esecuzione rapida e reattiva, il calcolo è parallelizzato tramite `Parallel.For` e sfrutta un sistema di *memory pooling* (`ArrayPool<double>.Shared`). Questa ottimizzazione riduce drasticamente l'impatto sul *Garbage Collector*, rendendo l'elaborazione fluida anche su immagini ad alta risoluzione. Infine, per evitare salti di intensità al raggio di taglio, il software calcola il valore del modello al bordo e applica una sfumatura gaussiana, garantendo una fusione naturale con il fondo cielo originale.

Un esempio dei risultati morfologici ottenibili tramite la sottrazione di questo modello sintetico, unitamente ai suoi fisiologici limiti di lieve sovra-sottrazione, è riportato nel pannello di confronto visivo di Figura 6.2.

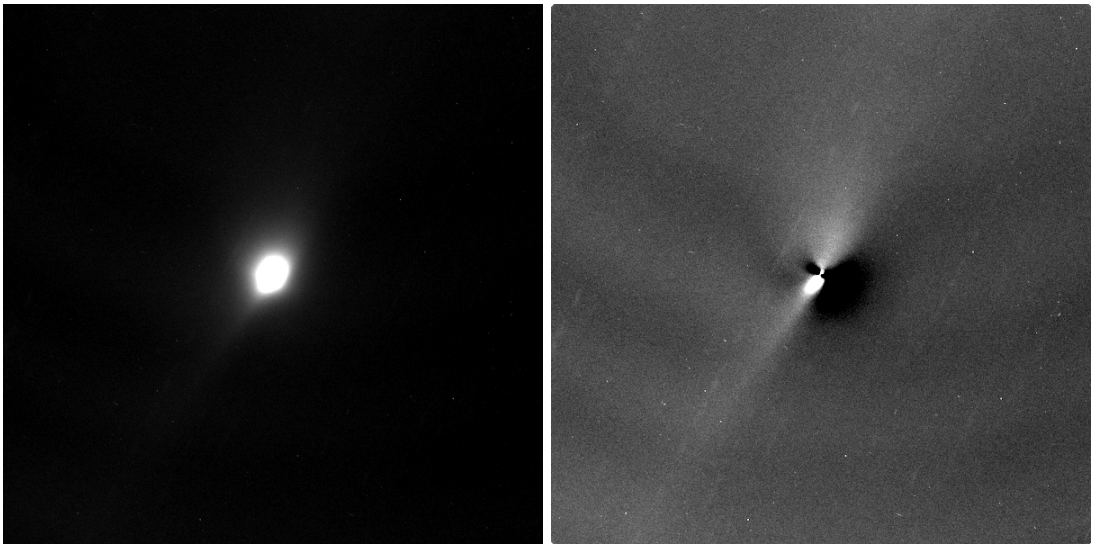

Figura 6.2. Risultato dell'applicazione del filtro MCM sulla cometa 67P. A sinistra, la ripresa di partenza dominata dal denso inviluppo diffuso; nel pannello di destra, l'esito della sottrazione del profilo sintetico mediano dal dato grezzo, operazione che svela ad alto contrasto le disomogeneità interne della chioma.

6.2.3 Radial Weighted Model (RWM)

Un approccio alternativo all'MCM, di natura puramente analitica, è costituito dal RWM. Questo algoritmo, derivato originariamente dalle ricerche di Tanyu Bonev e Klaus Jockers [48], condivide con i modelli precedenti l'obiettivo di rimuovere il velo fotometrico, ma lo fa basandosi su

un assunto fisico e non statistico: il flusso luminoso di una cometa incontaminata in espansione stazionaria si attenua proporzionalmente all'inverso della distanza dal nucleo ($\propto 1/\rho$) [47]. Basandosi su questo vincolo radiale, è possibile "appiattire" l'intero gradiente spaziale moltiplicando semplicemente l'intensità di ogni pixel per la sua distanza dal fotocentro.

Perché questa equazione risulti fisicamente valida, è tuttavia tassativo epurare il dato dal contributo costante del fondo cielo cosmico e del rumore termico del sensore. Il filtro si compone quindi di due fasi. La prima fase valuta il livello di fondo (B) campionando una griglia di regioni periferiche dell'immagine. Su questi campioni viene applicato un algoritmo iterativo di *Sigma-Clipping*: calcolata la mediana e la deviazione standard robusta, i pixel che si discostano eccessivamente (le stelle di fondo) vengono ripetutamente scartati, fino a far convergere il set di dati al reale valore del fondo cielo locale.

Nella seconda fase, l'algoritmo applica la correzione analitica pixel per pixel:

$$I_{RWM}(x, y) = [I(x, y) - B] \cdot \max(\rho(x, y), 1.0)$$

dove $\rho(x, y) = \sqrt{(x - x_c)^2 + (y - y_c)^2}$.

L'implementazione impone un limite matematico inferiore al raggio ($\max(\rho, 1.0)$): questo previene l'annullamento artificiale dell'intensità nel pixel coincidente con il centro esatto.

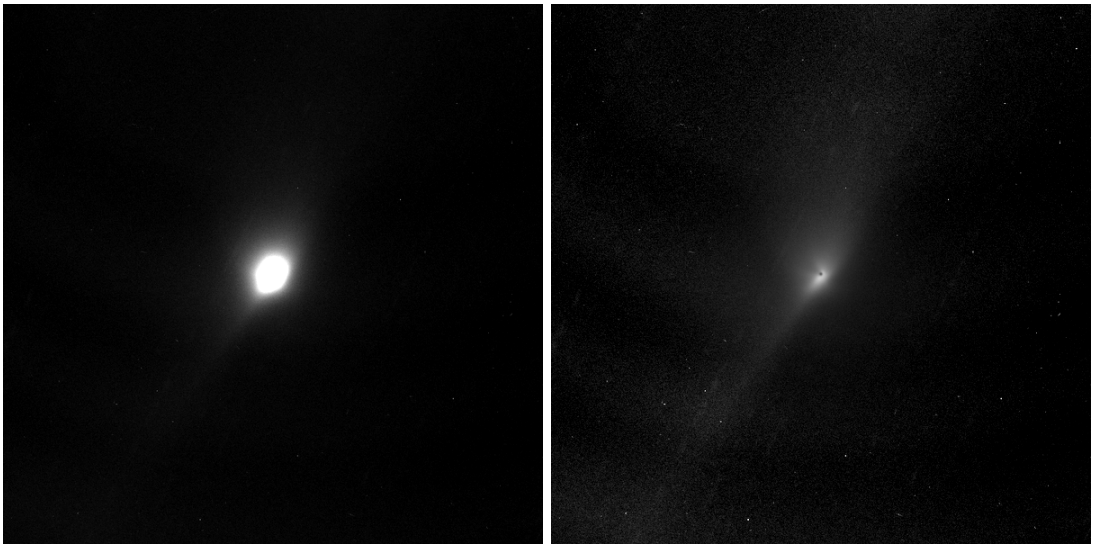

Figura 6.3. Confronto morfologico pre e post elaborazione RWM (cometa 67P). A sinistra, l'immagine di partenza fortemente condizionata dal decadimento $1/\rho$. A destra, il risultato della normalizzazione spaziale: la moltiplicazione per il raggio vettore, successiva alla decurtazione del fondo cielo via *Sigma-Clipping*, annulla il gradiente globale rivelando i dettagli a basso contrasto.

Dal punto di vista dell'ottimizzazione delle performance, le routine `RunRWMCore` evitano l'utilizzo di lenti iteratori bidimensionali sulla memoria non gestita. L'intera matrice viene preventivamente estratta in un array monodimensionale in virgola mobile (`GetArray`), permettendo a un costrutto `Parallel.For` di processare le righe in modo asincrono calcolando gli *offset* lineari. L'estrazione di invarianti dal ciclo più interno, come il pre-calcolo del quadrato della distanza verticale (dy^2), garantisce velocità di esecuzione ottimali anche su moli di dati considerevoli. Un esempio dell'isolamento delle strutture radiali ottenuto tramite l'appiattimento di questo gradiente medio è visibile in Figura 6.3.

6.2.4 Filtri di Normalizzazione Azimutale

A completamento delle tecniche basate sulla modellazione polare globale, si inserisce una famiglia di algoritmi sviluppata originariamente nell'ambito della *Cometary Coma Image Enhancement Facility* [49], basata sulla normalizzazione azimutale. A differenza dell'MCM che sottrae

un modello sintetico, queste tecniche operano come filtri divisivi o di *stretching* locale, agendo indipendentemente su ogni singola distanza radiale dal fotocentro.

L'algoritmo richiede preliminarmente la conversione dell'immagine dal dominio cartesiano a uno spazio polare ad alta risoluzione. Nell'implementazione di Kometra [3], questa complessa mappatura (`ToPolar`) è gestita preservando una rigorosa accuratezza *sub-pixel* e adottando un algoritmo di interpolazione *Lanczos-4*, essenziale per prevenire la perdita di micro-dettagli ad alta frequenza durante il campionamento circolare. Una volta traslata in coordinate polari, ogni riga della nuova matrice rappresenta l'insieme di tutti gli angoli azimutali per un dato raggio ρ.

Il software mette a disposizione tre varianti analitiche per il trattamento di queste righe:

- **Division by Azimuthal Average:** L'algoritmo calcola la media dei pixel lungo un raggio prefissato. Per evitare che getti luminosi, stelle di campo o raggi cosmici sfalsino il calcolo, viene applicato un pre-filtraggio statistico (*sigma-clipping*). I pixel che eccedono questa soglia di deviazione standard vengono scartati e la media viene ricalcolata in modo robusto. Infine, ogni pixel dell'anello viene diviso per questa media pulita.
- **Division by Azimuthal Median:** Variante concettualmente simile alla precedente, ma che impiega la mediana al posto della media. La differenza procedurale fondamentale rispetto all'MCM è che qui la divisione tra i valori originali e quelli trovati dalla mediana viene effettuata direttamente in coordinate polari; solo successivamente il risultato finale viene ritrasformato in coordinate cartesiane. Essendo la mediana un descrittore statistico intrinsecamente robusto agli *outlier*, questa variante omette la fase di *sigma-clipping*. È particolarmente indicata per immagini afflitte da densi campi stellari o pixel morti che potrebbero corrompere il calcolo della media.
- **Azimuthal Renormalization:** Invece di operare una divisione, questa tecnica esegue un vero e proprio *stretching* dinamico per ogni anello. Tramite un doppio passaggio di *sigma-clipping*, l'algoritmo identifica i limiti inferiore e superiore di intensità validi per un dato raggio ρ. I pixel dell'anello vengono quindi ri-normalizzati affinché la loro escursione riempia interamente un intervallo prefissato.

Sul fronte dell'elaborazione parallela, il software ottimizza queste trasformazioni estraendo vettori monodimensionali per ogni anello ed eseguendo i calcoli statistici tramite `Parallel.For`. Terminata la modulazione azimutale, la matrice polare subisce un *Warping* inverso (`FromPolar`). Per annullare gli artefatti da discontinuità al momento della riproiezione cartesiana, il software calcola mappe di coordinate esatte in virgola mobile e applica un margine virtuale ciclico (*Wrap padding*) in prossimità del raggio di taglio, restituendo una chioma in cui le variazioni angolari (i getti) risultano esaltate. Un esempio dell'efficacia di questa famiglia di algoritmi, ed in particolare della tecnica di *Azimuthal Renormalization*, è riportato nel pannello di confronto visivo di Figura 6.4.

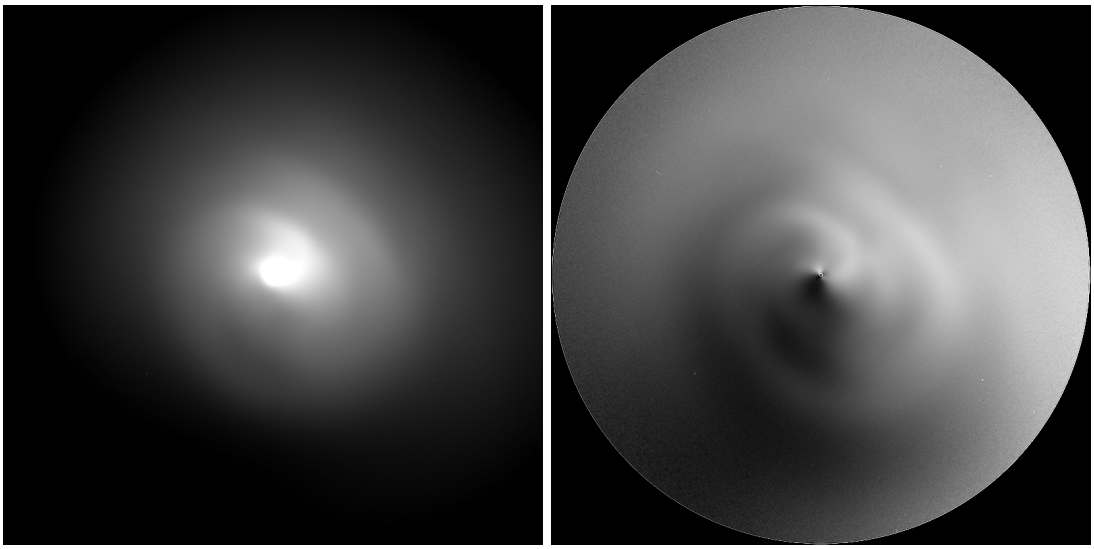

Figura 6.4. A sinistra, l'immagine originale in banda continua della cometa Hale-Bopp (dati acquisiti originariamente da D. Schleicher e L. Woodney, resi disponibili come dataset di test pubblico dal progetto CIEF [49]). A destra, il risultato dell'applicazione del filtro *Azimuthal Renormalization*. Trasformando temporaneamente l'immagine in coordinate polari, l'algoritmo ri-normalizza dinamicamente l'escursione luminosa per ogni singolo raggio di distanza dal nucleo. Questo "stretching" circolare annulla totalmente l'enorme gradiente radiale della chioma.

6.3 Normalizzazione Spaziale e Contrasto Locale

Mentre i filtri puramente geometrici mirano a sottrarre un modello globale della chioma parametrato dal nucleo, le tecniche di normalizzazione spaziale adottano un approccio locale. Queste metodologie si basano sull'assunto che le strutture cometarie di interesse (come filamenti o onde di densità) rappresentino variazioni ad alta frequenza spaziale rispetto al fondo immediatamente circostante. Operando su finestre mobili (*kernel*), gli algoritmi analizzano le proprietà statistiche del vicinato di ciascun pixel per esaltare i micro-contrasti, rendendoli indipendenti dal forte gradiente globale dell'oggetto celeste.

6.3.1 Filtro Spaziale a Varianza Radiale (RVSF)

Il RVSF è un operatore non lineare progettato specificamente per le comete, i cui algoritmi originali sono stati sviluppati da Nalin Samarasinha e Steve Larson [49]. A differenza dei classici filtri di convoluzione che utilizzano un *kernel* di dimensioni fisse, il RVSF adatta dinamicamente la dimensione della propria finestra di campionamento in funzione della distanza dal nucleo. Nei pressi del centroide, dove i dettagli sono minuti, il filtro opera su una finestra ristretta; nelle regioni esterne della chioma, per compensare la dispersione delle velocità e le maggiori dimensioni delle strutture, la finestra si allarga.

Per ogni pixel bersaglio, l'algoritmo valuta un gruppo di 8 pixel di riferimento disposti su un quadrato: i quattro vertici e i quattro punti medi dei lati. La distanza a tra il pixel centrale e i pixel di bordo è governata dalla legge di potenza:

$$a = A + B \cdot \rho^N$$

dove ρ è la distanza dal fotocentro. I parametri empirici giocano ruoli distinti: A controlla la dimensione del *kernel* nella chioma interna, N (il termine esponenziale) governa l'espansione del *kernel* nella chioma esterna, mentre B funge da fattore di scala lineare globale. Inoltre, la *Facility* originale prevede la possibilità di applicare preliminarmente un logaritmo al valore dei pixel per comprimere la gamma dinamica (*log option*).

Prendendo come base di studio il codice sorgente originale in linguaggio Fortran reso pubblico dagli autori sul portale della *Facility* [49], l'implementazione in C# nativo sviluppata in Kometra [3] traduce e ottimizza l'operatore differenziale in un'espressione algebrica pesata ad alte prestazioni:

$$I_{RVSF}(x,y) = 1024 \cdot I(x,y) - 192 \cdot \Sigma I_{ep} - 64 \cdot \Sigma I_{cp}$$

dove $I(x,y)$ è il valore del pixel centrale, ΣI_{ep} è la somma dei quattro pixel ai bordi, e ΣI_{cp} è la somma dei quattro pixel ai vertici. Questa formulazione matematica fa sì che nelle regioni a luminosità omogenea il risultato si annulli termicamente a zero, esaltando per contrasto le variazioni anomale (i getti).

Durante lo sviluppo della routine, l'adozione di coordinate rigorosamente *sub-pixel* per il calcolo della distanza ρ ha permesso di annullare gli artefatti da quantizzazione al centro del nucleo. Inoltre, il calcolo dinamico dei limiti del *kernel* (`yMin`, `yMax`, ecc.) impedisce violazioni di accesso alla memoria quando la finestra di campionamento eccede i bordi fisici dell'immagine.

Trovare la combinazione ideale dei parametri A, B e N per una specifica immagine richiede tipicamente numerosi tentativi empirici. Per snellire questo processo analitico, Kometra [3] implementa anche la variante definita *RVSF Mosaic*. L'utente fornisce due valori limite per ciascun parametro, e il motore genera una matrice di 8 combinazioni distinte, calcolate simultaneamente su un'unica tavola visiva. A livello architetturale, questa operazione *memory-intensive* richiede di allocare un'immagine di destinazione avente dimensioni pari a 4×2 volte l'originale, su cui vengono mappate le 8 riproiezioni. Per prevenire la saturazione della memoria RAM e il collasso dell'applicativo, il lancio dei *Task* paralleli è strettamente regolato da un `SemaphoreSlim` . Questo costrutto di sincronizzazione asincrona garantisce che il *Garbage Collector* di .NET non venga mai sopraffatto dall'allocazione simultanea di matrici massive, rendendo la generazione del mosaico stabile e reattiva.

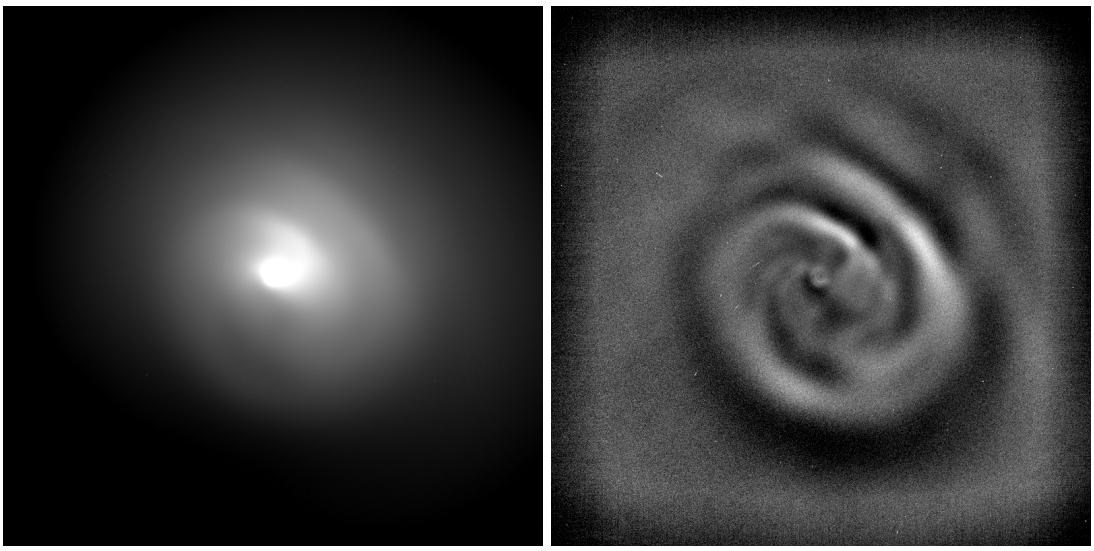

Figura 6.5. Analisi comparativa dell'elaborazione tramite algoritmo RVSF. A sinistra: l'immagine originale in banda continua (emissione delle polveri) della cometa Hale-Bopp [49]. A destra: l'esito dell'applicazione del filtro spaziale.

6.3.2 Normalizzazione Spaziale Locale (LSN) Adattiva

Un'evoluzione logica per generalizzare l'approccio locale è costituita dalla Normalizzazione Spaziale Locale (LSN). Questo filtro standardizza statisticamente ogni pixel dell'immagine rispetto al suo vicinato, trasformando la matrice originale in una mappa delle deviazioni standard locali:

$$I_{LSN}(x,y) = \frac{I(x,y) - \mu_{local}}{\sigma_{local} + \epsilon}$$

dove μ_{local} e σ_{local} rappresentano rispettivamente la media e la deviazione standard all'interno del *kernel* centrato in (x, y), mentre ϵ è una costante infinitesima (impostata a 10^{-9} nell'implementazione) introdotta per prevenire instabilità matematiche o divisioni per zero in regioni a varianza nulla.

La criticità maggiore nell'implementazione di questo algoritmo in ambito astronomico risiede nella presenza di dati invalidi. A seguito delle operazioni di allineamento geometrico o di mascheramento delle stelle sature, le matrici FITS contengono frequentemente pixel di valore NaN. I tradizionali filtri di convoluzione basati su trasformate di Fourier o su finestre scorrevoli sono vulnerabili a tale condizione: un singolo NaN all'interno del *kernel* si propaga, invalidando l'intero blocco spaziale circostante.

Per ovviare a questo problema, si è ingegnerizzata un'architettura definita *NaN-Safe*. Invece di iterare manualmente sui pixel, l'algoritmo separa concettualmente il dato dalla sua validità. La routine estrae l'immagine originale in una matrice pulita (dove i NaN vengono declassati a valore numerico zero) e genera parallelamente una mappa dei pesi (*weight map*), contenente il valore 1.0 per i pixel validi e 0.0 per i pixel corrotti.

Sfruttando l'elevata efficienza delle convoluzioni tramite *Filtro Gaussiano* di OpenCV, l'algoritmo calcola le somme pesate locali sia sulla matrice pulita sia sulla mappa dei pesi. L'adozione di una campana Gaussiana in luogo di un tradizionale filtro rettangolare (*Box Filter*) risulta cruciale per garantire transizioni spaziali morbide e prevenire la formazione di artefatti a blocchi. Dividendo il risultato della prima convoluzione per la seconda, si ottiene il valore esatto di μ_{local} calcolato esclusivamente sui pixel validi sopravvissuti all'interno della finestra. La medesima logica viene applicata ai quadrati dei valori per l'estrazione sicura della varianza ($\sigma^2_{local} = \mu_{x^2} - \mu^2_x$). Questa elegante formulazione algebrica produce inoltre un effetto di *inpainting* naturale: persino in corrispondenza di un pixel originariamente mascherato, l'equazione calcola un valore numerico continuo basato sul vicinato sano, scongiurando la propagazione dell'errore.

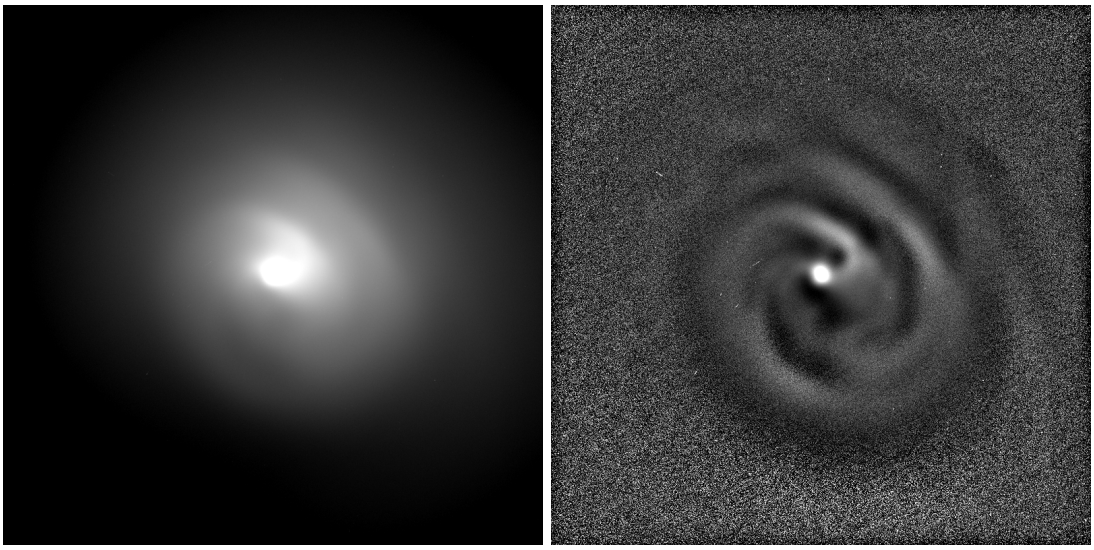

Figura 6.6. Confronto visivo dell'elaborazione tramite LSN. A sinistra, l'immagine originale della cometa Hale-Bopp [49]. A destra, il risultato dell'algoritmo. La standardizzazione statistica del vicinato permette di appiattire le macro-variazioni di luminosità, portando in risalto i micro-contrasti e i getti locali. Grazie all'utilizzo di convoluzioni sfumate e della mappa dei pesi, l'algoritmo non genera artefatti a blocchi nemmeno in prossimità di bruschi gradienti o di pixel precedentemente mascherati.

Infine, poiché il calcolo puro produce un tensore di scostamenti statistici privo del range dinamico originale, il motore applica un fattore di scala (*Intensity*) governato dall'utente. Questo parametro permette di modulare l'iniezione del contrasto locale all'interno dell'immagine di base, garantendo un blending naturale. L'intero approccio assicura una perfetta inalterabilità del rigore scientifico anche ai bordi più estremi del campo visivo o in prossimità di maschere stellari molto

dense. Un esempio del recupero di informazione ad alta frequenza tramite LSN è visibile in Figura 6.6.

6.3.3 Unsharp Masking Non Lineare

A completamento della suite di contrasto locale, il motore integra una tecnica classica di separazione delle frequenze. L'*Unsharp Masking* tradizionale sottrae all'immagine originale una sua versione sfocata (solitamente tramite operatore Gaussiano), agendo di fatto come un filtro passa-alto. Per l'analisi morfologica cometaria si è optato per una variante non lineare: l'*Unsharp Masking Mediano*. L'impiego di un filtro mediano in sostituzione di quello gaussiano per la generazione della maschera sfocata mira a preservare l'integrità dei gradienti di intensità elevati e i profili netti dei getti, mitigando la diffusione dell'informazione spaziale tipica degli operatori lineari. L'operazione è definita matematicamente come:

$$I_{UM} = I(x, y) - \text{Mediana}(I, k)$$

Sul fronte della gestione della memoria e del carico computazionale, il calcolo della mediana su finestre scorrevoli bidimensionali costituisce un'operazione fortemente critica. Il software adotta pertanto una logica di *dispatching* dinamico: per *kernel* piccoli ($k \leq 5$) si appoggia alle routine ottimizzate in C++ della libreria OpenCV (`Cv2.MedianBlur`), mentre per raggi d'azione elevati devia il flusso esecutivo su un'implementazione multithread proprietaria. Per supportare *kernel size* grossi senza esaurire la memoria temporanea, questa routine sfrutta un'architettura basata su `ThreadLocal`, strettamente governata dal semaforo di memoria globale per impedire il collasso della RAM in fase di allocazione dei *buffer* di ordinamento.

Nonostante l'ottimizzazione architetturale, i test empirici hanno dimostrato che l'utilità pratica di questo operatore in ambito cometario è marginale. La complessità computazionale dell'algoritmo di ordinamento statistico cresce in modo non lineare rispetto alla dimensione della finestra, rendendo l'esecuzione estremamente lenta per i raggi estesi necessari a isolare le macro-strutture della chioma. Inoltre, dal punto di vista prettamente morfologico, il risultato sottrattivo non ha mostrato vantaggi qualitativi o quantitativi di rilievo rispetto alla pulizia e all'efficienza garantite dalle tecniche analizzate nelle sezioni precedenti.

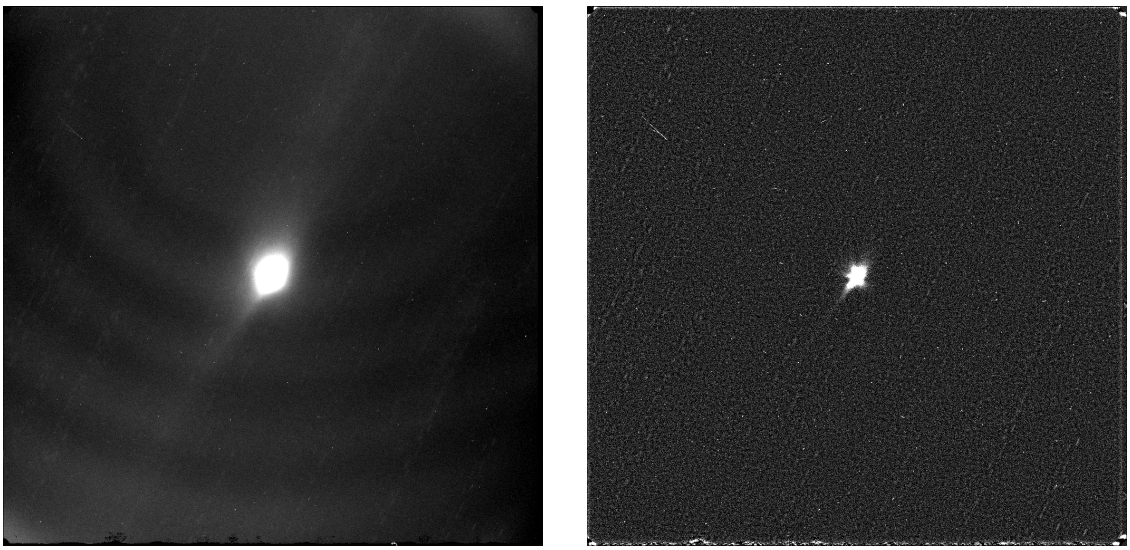

Figura 6.7. Risultato dell'applicazione dell'*Unsharp Masking Mediano*. Rispetto all'immagine originale della cometa 67p (a sinistra), il filtro estrae le alte frequenze spaziali (a destra). Tuttavia, dal punto di vista analitico, il risultato non giustifica l'enorme costo computazionale richiesto dall'ordinamento statistico per grandi finestre di campionamento, risultando inferiore per pulizia e controllo rispetto agli operatori differenziali puri.

6.3.4 Contrast Limited Adaptive Histogram Equalization (CLAHE)

A valle delle elaborazioni sulle frequenze locali, l'estrazione visiva finale dei dettagli è solitamente affidata ad algoritmi di *stretching* dell'istogramma. Il CLAHE [50], algoritmo sviluppato originariamente nell'ambito dell'*imaging* diagnostico per esaltare i tessuti molli nelle radiografie senza saturare le strutture ossee, rappresenta un'evoluzione sofisticata della normale equalizzazione globale. Operando su una griglia di tasselli indipendenti (*tiles*), equalizza il contrasto localmente e impone un limite matematico alla pendenza della curva di trasferimento (*Clip Limit*), con lo scopo teorico di amplificare i debolissimi dettagli della chioma senza saturare artificialmente il nucleo.

In fase di prototipazione, per valutare l'efficacia morfologica di questo operatore prima di investire risorse nello sviluppo di un complesso algoritmo *custom* nativo in virgola mobile, si è optato per un'implementazione di test basata sulle routine standard della libreria OpenCV. Tale scelta ha comportato un compromesso tecnico consapevole: poiché OpenCV non supporta l'esecuzione del CLAHE su matrici FITS ad alta precisione, la pipeline esegue un *casting* temporaneo, mappando l'immagine nel dominio discreto a 16 bit (range $0 - 65535$). Si era dunque coscienti che questa discretizzazione avrebbe introdotto un errore di quantizzazione, troncando le preziose variazioni fotometriche sub-decimali, ma tale perdita di precisione è stata ritenuta inizialmente accettabile ai soli fini esplorativi del test.

Tuttavia, le analisi empiriche hanno rapidamente dimostrato che i limiti di questo approccio vanno ben oltre la mera perdita di precisione numerica, portando alla decisione di scartare definitivamente questa strada. Il difetto strutturale insormontabile risiede nella gestione delle frequenze spaziali: nel tentativo di far emergere le striature interne più fini, l'operatore richiede di ridurre progressivamente la dimensione della cella di analisi. Su gradienti complessi ed estremamente ripidi come quelli cometari, l'interpolazione bilineare incaricata di smussare i confini tra i tasselli fallisce, innescando evidenti artefatti geometrici. La sovrapposizione visiva di questa griglia artificiale a scacchiera inquina in modo inaccettabile il dato fisico, rendendo impossibile distinguere i reali getti di polvere dai bordi di calcolo dell'algoritmo.

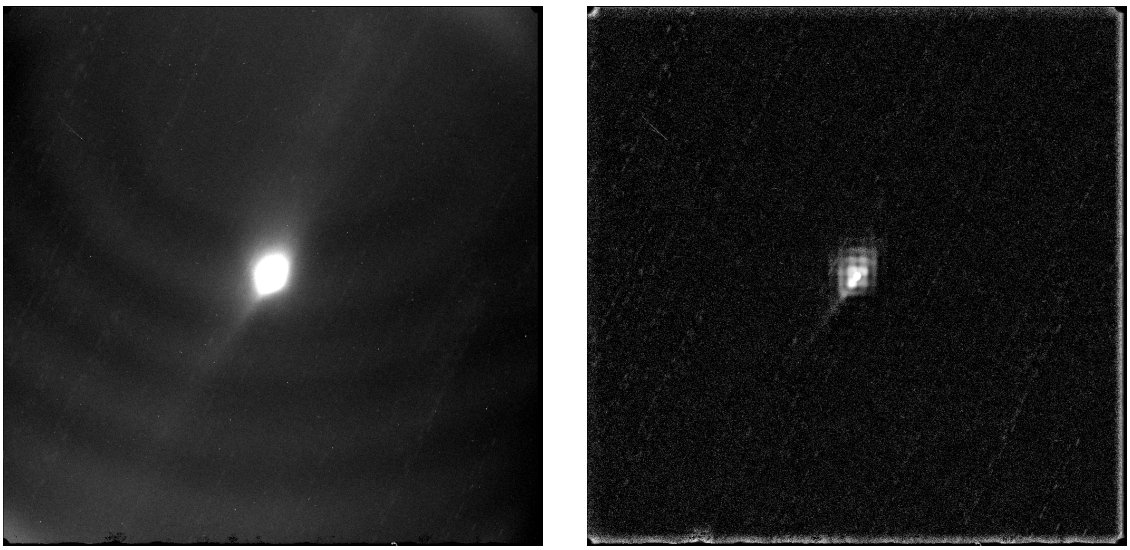

Figura 6.8. Criticità nell'elaborazione tramite CLAHE su profili cometari a elevata dinamica. Nel pannello di destra, l'esasperazione dell'equalizzazione dell'istogramma, volta a estrarre i dettagli minuti della chioma, evidenzia le vulnerabilità strutturali del metodo. Il collasso dell'interpolazione bilineare genera evidenti artefatti a scacchiera (*grid aliasing*) che alterano irrimediabilmente la morfologia fisica della polvere cometaria, invalidandone lo studio.

6.4 Operatori Differenziali e Analisi Tensoriale

I filtri radiali e di normalizzazione locale analizzati sinora si dimostrano estremamente efficaci nell'isolare le alte frequenze, ma non possiedono alcuna "consapevolezza" della forma geometrica intrinseca della struttura che stanno elaborando. Per l'identificazione esplicita di morfologie cometarie complesse, quali i sottili e allungati getti di polvere, Kometra [3] integra un motore di analisi dedicato che sfrutta le derivate spaziali e i tensori per misurare matematicamente la pendenza, la curvatura e l'anisotropia del campo luminoso. Questa sezione documenta un'evoluzione progressiva: partendo dai limiti delle derivate seconde isotrope, si esplorerà l'applicabilità di complessi tensori direzionali mutuati dalla geofisica e dalla diagnostica medica.

6.4.1 Operatore Laplaciano Adattivo

L'operatore Laplaciano rappresenta il fondamento matematico dell'analisi differenziale isotropa. In termini fisici, esso calcola la divergenza del gradiente di luminosità dell'immagine, misurando la derivata parziale del secondo ordine non direzionale:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

L'applicazione di questo operatore su un'immagine astrofisica mira a evidenziare le regioni di massima concavità o convessità del profilo luminoso, isolando i fronti ad alta frequenza dal gradiente radiale continuo della chioma, il quale, variando in modo dolce e progressivo, viene matematicamente appiattito verso lo zero.

Nel contesto dello sviluppo di Kometra [3], l'implementazione diretta di tale derivata su dati grezzi si scontra con un limite ben noto nella letteratura informatica: essendo un filtro passa-alto del secondo ordine, il Laplaciano amplifica in modo quadratico il rumore termico del sensore e le variazioni puntiformi del fondo cielo, rendendo tipicamente illegibile il segnale della cometa.

Per arginare questo fenomeno senza compromettere la risoluzione spaziale, la routine implementata introduce una rigorosa soluzione architetturale: prima dell'estrazione differenziale, l'immagine, mantenuta tassativamente in formato a virgola mobile ad alta precisione, viene sottoposta a un filtraggio SNN. A differenza della tradizionale sfocatura Gaussiana (*Laplacian of Gaussian* - LoG), la quale tende a "spalmare" e degradare i confini delle deboli strutture cometarie, il pre-filtraggio SNN abbatte la varianza del rumore di fondo preservando intatta la ripidità dei gradienti locali.

A valle di questa regolarizzazione, l'operazione differenziale viene eseguita tramite una convoluzione spaziale. Al fine di rispettare le prassi analitiche astrofisiche, i segni dell'operatore standard sono stati matematicamente invertiti:

$$K_{Laplace} = \begin{bmatrix} -0.125 & -0.125 & -0.125 \\ -0.125 & 1.0 & -0.125 \\ -0.125 & -0.125 & -0.125 \end{bmatrix}$$

Tramite questa configurazione algoritmica personalizzata, le emissioni fisiche (come i getti di polvere) producono nativamente valori derivativi positivi, emergendo come anomalie strutturali ad alto contrasto. Parallelamente, l'utilizzo delle matrici `CV_32F` o `CV_64F` previene il troncamento (*clipping*) sullo zero termico, preservando in memoria l'escursione negativa delle fisiologiche "valli" oscure di compensazione (*ringing*) che si generano ai margini dei forti sbalzi luminosi.

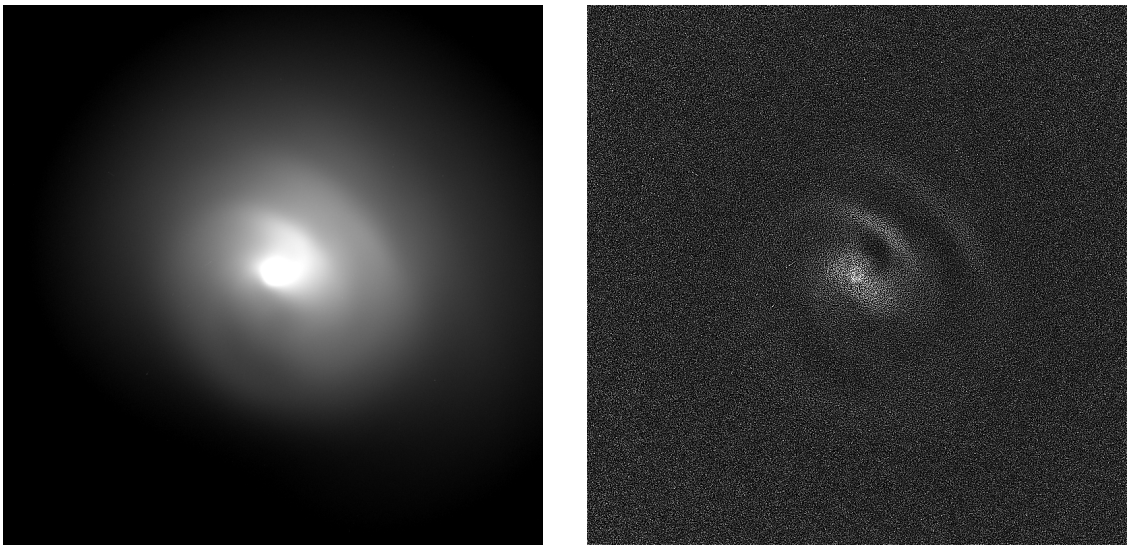

Figura 6.9. Elaborazione morfologica tramite Operatore Laplaciano Adattivo. A sinistra, l'immagine originale [49] dominata dall'intensità del nucleo. A destra, il risultato dell'estrazione differenziale del secondo ordine: grazie all'azione stabilizzante del pre-filtraggio SNN e all'inversione dei segni del *kernel*, i fronti d'onda e le discontinuità strutturali della chioma vengono isolati con successo e trascritti come valori positivi, rimuovendo integralmente la componente radiale a bassa frequenza spaziale.

6.4.2 Filtro di Frangi (Vesselness)

Quale evoluzione per arginare la natura isotropa delle derivate di base, il filtro di Frangi, concepito originariamente in ambito biomedico per l'enfatizzazione dei vasi sanguigni nelle angiografie [51], è stato integrato in Kometra [3] a scopo sperimentale. L'ipotesi di lavoro prevedeva di modellare topologicamente un getto collimato emesso dal nucleo come una struttura tubolare o filamentosa, immersa in un fondo diffuso.

Per quantificare la probabilità che un pixel appartenga a una struttura filamentosa direzionata, l'algoritmo valuta le derivate parziali del secondo ordine dell'immagine $I(x, y)$, costruendo la matrice Hessiana H per ogni punto:

$$H = \begin{bmatrix} L_{xx} & L_{xy} \\ L_{xy} & L_{yy} \end{bmatrix}$$

Per rendere l'operatore sensibile unicamente alla scala spaziale della struttura desiderata, le derivate non vengono calcolate sull'immagine cruda, bensì a valle di una convoluzione con un nucleo Gaussiano di deviazione standard σ configurabile.

Estratta la matrice Hessiana, il sistema ne calcola analiticamente gli autovalori λ_1 e λ_2 (ordinati tali per cui $|\lambda_1| \leq |\lambda_2|$). In una struttura tubolare ideale, la curvatura è trascurabile lungo la direzione del filamento ($\lambda_1 \approx 0$), ma risulta estremamente convessa nella direzione ortogonale ($\lambda_2 \ll 0$). La misura della "tubolarità" (*Vesselness*) è data dalla formula non lineare:

$$V(x, y) = \begin{cases} 0 & \text{se } \lambda_2 \geq 0 \\ \exp\left(-\frac{R_B^2}{2\beta^2}\right) \cdot \left(1 - \exp\left(-\frac{S^2}{2c^2}\right)\right) & \text{altrimenti} \end{cases}$$

dove $R_B = |\lambda_1|/|\lambda_2|$ distingue le forme intrinsecamente simmetriche dal rumore, $S = \sqrt{\lambda_1^2 + \lambda_2^2}$ sopprime il fondo a bassa intensità, mentre β e c sono soglie di sensibilità configurabili.

Nella traduzione algoritmica di questo modello matematico, il calcolo di gradienti di secondo ordine su transizioni luminose estreme espone il software a severi rischi numerici. La routine C# converte temporaneamente il tensore in doppia precisione a 64 bit, ricavando le derivate spaziali

tramite operatori morfologici di Sobel. Per garantire prestazioni adeguate, la diagonalizzazione dell'Hessiana è risolta all'interno di un *kernel* nativo parallelizzato e protetto da semafori di memoria.

Nonostante l'accuratezza teorica e l'ottimizzazione del codice, i test empirici hanno rivelato che i risultati ottenuti in ambito cometario sono, nella maggior parte dei casi, insufficienti. Ad eccezione di macro-strutture di dimensioni notevoli, l'operatore fatica tipicamente a estrarre informazioni morfologiche di reale interesse scientifico. Il limite principale risiede nella forte dipendenza da parametri operativi (σ, β, c) la cui calibrazione risulta estremamente complessa e poco generalizzabile. A questo si aggiunge la rigidità del modello geometrico di base ($\lambda_1 \approx 0$), che mal si adatta alla naturale curvatura assunta dai getti a causa della rotazione del nucleo, tendendo ad amplificare il rumore di fondo a discapito della continuità strutturale. Questo comportamento, caratterizzato dalla mancata estrazione delle strutture d'interesse e dall'esplosione del rumore di base, è chiaramente documentato in Figura 6.10.

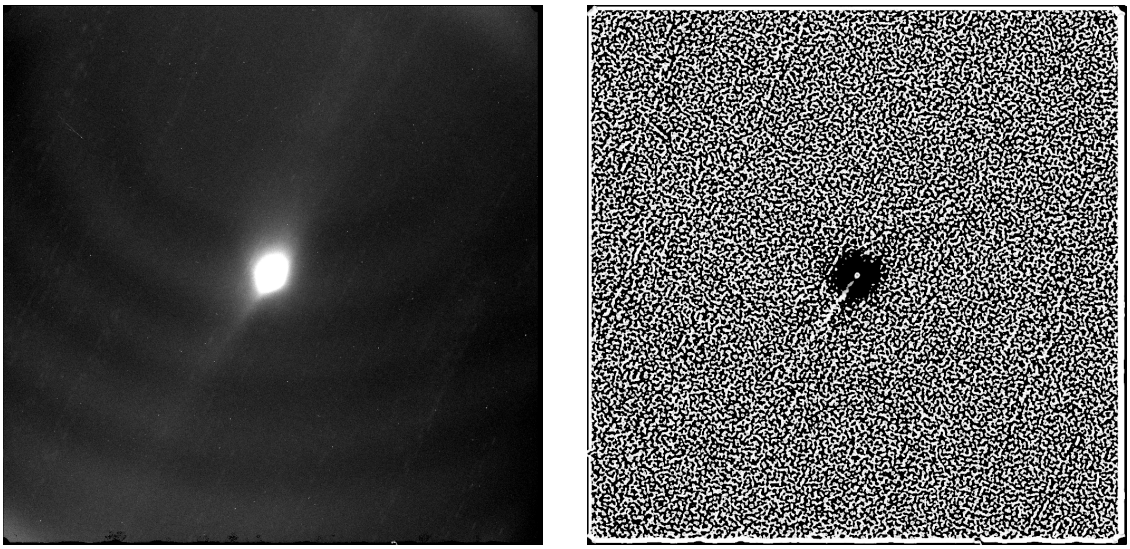

Figura 6.10. Limiti operativi del filtro di Frangi applicato alla morfologia cometaria. Rispetto all'immagine di partenza della cometa 67p (a sinistra), l'analisi Hessiana (a destra) fallisce nell'estrazione dei deboli getti curvi, i quali non rispettano il rigido vincolo geometrico di tubolarità ($\lambda_1 \approx 0$). L'effetto collaterale più evidente e distruttivo è la drastica amplificazione del rumore termico e di fondo.

6.4.3 Tensore di Struttura

A parziale superamento delle fragilità riscontrate nell'analisi Hessiana, e nel tentativo di arginare la sensibilità al rumore tipica delle derivate del secondo ordine, si è testato un operatore mutuato dalla *Computer Vision*: il Tensore di Struttura [52]. Originariamente introdotto per il tracciamento del flusso ottico e dei bordi nelle immagini digitali, e successivamente adottato in sismologia per mappare i fronti d'onda nel sottosuolo, questa tecnica valuta la coerenza e l'anisotropia direzionale basandosi unicamente sul gradiente del primo ordine. Lo scopo teorico è identificare fronti d'onda e gradienti a gradino (*step-edges*) bypassando il vincolo di tubolarità.

Il Tensore di Struttura J si ottiene calcolando i prodotti esterni del vettore gradiente spaziale $\nabla I = [I_x, I_y]^T$ e operandone un'integrazione locale su una finestra definita da un filtro Gaussiano di scala ρ:

$$J = G_\rho * \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Anche in questo dominio, l'estrazione degli autovalori μ_1 e μ_2 della matrice simmetrica J permette di definire la coerenza locale dell'immagine. La metrica di *Coherence* C viene calcolata come:

$$C = \left(\frac{\mu_1 - \mu_2}{\mu_1 + \mu_2} \right)^2$$

Il risultato di questa equazione produce una mappa di modulazione confinata nell'intervallo $[0, 1]$, ideata per sopprimere il segnale nelle aree caotiche ed enfatizzare unicamente le regioni caratterizzate da un forte allineamento strutturale.

Per ottimizzare il flusso di calcolo, l'implementazione C# processa l'estrazione valutando le derivate prime tramite l'operatore di Sobel, per poi generarne i prodotti quadratici. L'integrazione spaziale è demandata a passate di sfocatura Gaussiana. Analogamente al filtro di Frangi, l'algoritmo protegge le allocazioni di memoria tramite semaforo e devia l'estrazione tensoriale verso un *kernel* nativo parallelizzato, calcolato rigorosamente in virgola mobile per preservare il dato fotometrico.

Nonostante la robustezza teorica e programmatica dell'implementazione, i test empirici hanno dimostrato che anche il Tensore di Struttura risulta inadeguato per l'analisi cometaria, fornendo risultati inferiori alle aspettative. Il fallimento dell'operatore è da imputare alla natura stessa del campo vettoriale della chioma: il gradiente spaziale primario (∇I) di una cometa è dominato in modo schiacciante dall'escursione radiale isotropa della luminosità (il decadimento dal nucleo verso il fondo cielo). Il Tensore di Struttura viene letteralmente "abbagliato" da questo macro-gradiente radiale, mappando la forma sferica della chioma ma risultando completamente cieco di fronte alle micro-variazioni di coerenza direzionale indotte dai getti di polvere.

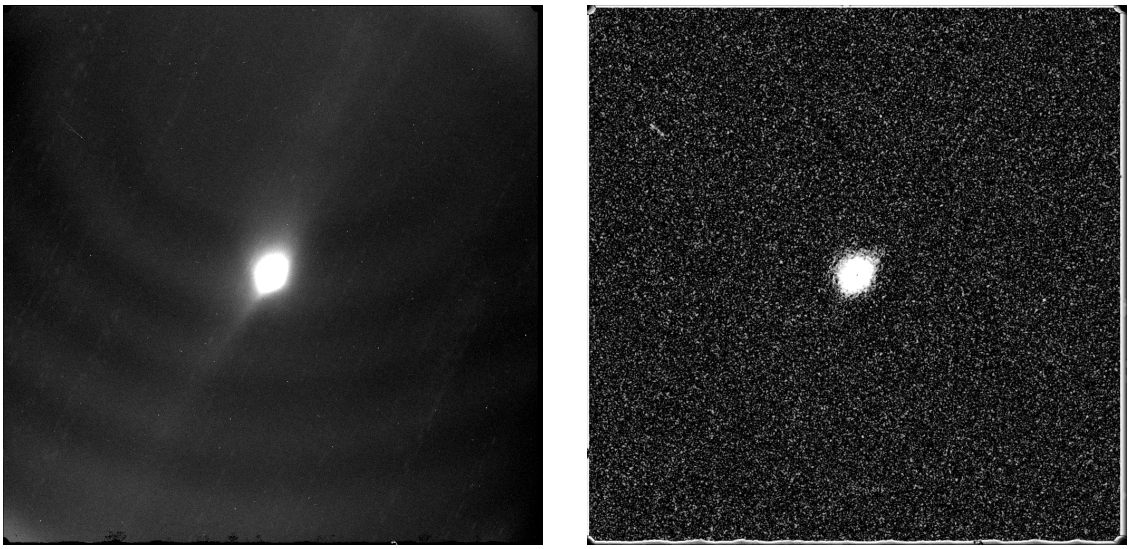

Figura 6.11. Inefficacia del Tensore di Struttura nell'isolamento dei dettagli cometari. Confrontando l'immagine originale (a sinistra) con la mappa di coerenza direzionale estratta (a destra), risulta evidente come l'algoritmo venga completamente saturato dal macro-gradiente radiale della chioma. L'enorme escursione luminosa dal nucleo verso l'esterno "abbaglia" la rilevazione delle derivate prime, impedendo all'operatore di percepire e isolare le deboli anisotropie locali generate dai getti di polvere.

6.5 Operatori Morfologici

A distaccarsi dalle operazioni algebriche o differenziali sin qui descritte, Kometra [3] integra un modulo dedicato alla manipolazione diretta delle forme geometriche presenti nell'immagine, basato sui principi della Morfologia Matematica.

6.5.1 White Top-Hat

L'ultimo strumento d'indagine fornito dal modulo è l'operatore morfologico *White Top-Hat*. Nata negli anni '80 per applicazioni di mineralogia e conteggio cellulare in biologia, questa tecnica viene impiegata tradizionalmente per estrarre elementi chiari che possiedano dimensioni inferiori a quelle di un elemento strutturante prefissato.

A livello formale, il White Top-Hat T_w è definito come la differenza algebrica tra l'immagine originale I e la sua apertura morfologica (erosione seguita da dilatazione) con un elemento strutturante S:

$$T_w(I) = I - (I \circ S)$$

Nel contesto teorico dell'analisi cometaria, dimensionando l'elemento strutturante in modo che sia marginalmente più ampio della sezione trasversale dei getti di polvere, l'apertura morfologica dovrebbe cancellare fisicamente i getti, lasciando intatta l'ampia e dolce curvatura della chioma di fondo. Sottraendo questa stima dall'immagine di partenza, i getti riapparirebbero con un altissimo contrasto su un fondo idealmente nullo.

Nell'eseguire questa sequenza, la routine C# applica preventivamente una leggera sfocatura Gaussiana (*pre-smoothing*) per mitigare l'interferenza del rumore puntiforme, per poi istanziare un elemento strutturante ellittico di dimensione configurabile. L'operazione morfologica viene delegata ai *kernel* ottimizzati di OpenCV, assicurando infine il troncamento a zero (*clipping*) di eventuali valori negativi residui.

Tuttavia, l'applicazione pratica di questa tecnica sui reali dati FITS ha portato a esiti fuorvianti, decretandone la totale inapplicabilità a fini di ricerca. Il limite critico risiede nell'interazione tra l'elemento strutturante a geometria fissa e l'estremo gradiente radiale della cometa. L'apertura morfologica su un gradiente così ripido non produce uno sfondo liscio, ma innesca un grave fenomeno di "terrazzamento" (*terracing*), generando uno sfondo a scalini artificiali. La sottrazione di questo fondo produce artefatti strutturali gravissimi: la trasformazione tende letteralmente a creare falsi dettagli luminosi, sotto forma di archi o striature, indotti unicamente dalla matrice dell'algoritmo.

L'aspetto più insidioso e scientificamente inaccettabile di questo operatore è che tali artefatti matematici assumono forme che simulano la morfologia dei reali getti di polvere o dei gusci concentrici (*shells*). Per questo motivo, l'impiego di questa tecnica invalida completamente e irrimediabilmente lo studio morfologico, rischiando di indurre il ricercatore a misurare e catalogare strutture inesistenti.

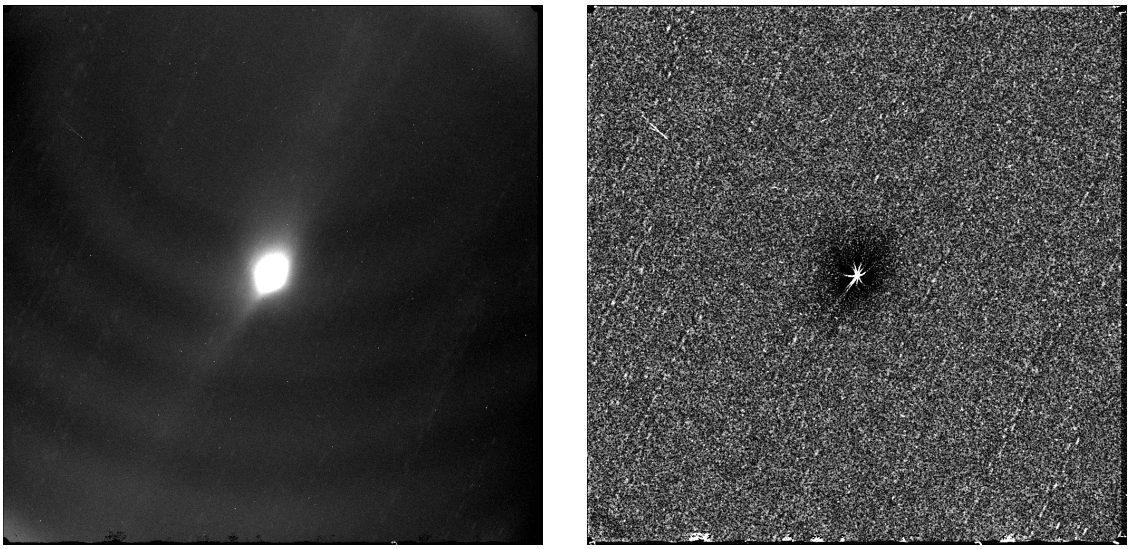

Figura 6.12. Generazione di falsi positivi strutturali tramite operatore *White Top-Hat*. L'immagine risultante (a destra), ottenuta applicando il filtro all'originale (a sinistra), mostra l'estrazione di evidenti morfologie arcuate e filamentose. Tuttavia, l'analisi critica rivela che si tratta di puri artefatti matematici indotti dall'interazione tra l'elemento strutturante ellittico e il ripido gradiente sferico della chioma. L'impossibilità di distinguere visivamente queste strutture fittizie dai reali getti cometari invalida completamente l'uso scientifico di questa tecnica morfologica.

6.6 Architettura Software: Motori di Filtraggio e Ottimizzazione

Come per le funzioni già analizzate nei capitoli precedenti, l'architettura di *backend* si biforca in due macro-livelli operativi, separando nettamente i motori di calcolo puro (*Engines*) dai moduli di orchestrazione del flusso dati (*Coordinators*).

Gli *Engines* operano in uno stato di totale isolamento: accettano e restituiscono matrici di pixel (`OpenCvSharp.Mat`), ignorando deliberatamente l'esistenza del file system o della struttura FITS. Per rispecchiare la tassonomia matematica degli algoritmi analizzati, questo strato è stato declinato in classi specializzate indipendenti. La classe `GradientRadialEngine` incapsula la logica delle trasformazioni cartesiano-polari e della modellazione globale, gestendo l'applicazione dei filtri di Larson-Sekanina, dell'MCM, del RWM e delle normalizzazioni azimutali. Le operazioni statistiche su finestre mobili, come la Normalizzazione Spaziale Locale (LSN), il CLAHE e l'*Unsharp Masking*, sono invece demandate alla classe `LocalContrastEngine`. Per i calcoli differenziali pesanti, la logica è stata isolata all'interno dello `StructureShapeEngine`, che ospita le routine per l'Hessiana di Frangi, il Tensore di Struttura, l'estrazione morfologica *Top-Hat* e il Laplaciano Adattivo.

Per aggirare i colli di bottiglia tipici del *framework* .NET nella manipolazione di tensori non gestiti, il codice di questi motori è stato sottoposto a un'intensa ottimizzazione. L'elaborazione spaziale sfrutta il multithreading tramite costrutti `Parallel.For`, mentre l'accesso inter-pixel avviene mediante iteratori generici (`GetGenericIndexer<T>`). Questa scelta garantisce velocità di esecuzione paragonabili all'uso di puntatori *unsafe* in C/C++, mitigando al contempo i rischi di violazione di memoria. Sul fronte della gestione delle risorse, algoritmi *memory-bound* soggetti a continue allocazioni temporanee aggirano la pressione sul *Garbage Collector* sfruttando pool di memoria condivisi, come `ArrayPool<T>.Shared`, o *buffer* pre-allocati tramite `ThreadLocal<T>`. Inoltre, per prevenire la saturazione della RAM (*OutOfMemoryException*) durante le massicce estrazioni tensoriali, il flusso di *Task* asincroni è rigorosamente governato da costrutti `SemaphoreSlim`, calibrati dinamicamente sui *core* fisici del sistema.

Al livello superiore, l'orchestrazione dell'intera *pipeline* logica e funzionale è affidata al *Dispatcher* centrale, nello specifico l'`ImageEnhancementCoordinator`. Iniettando al proprio interno i motori matematici e i gestori di I/O, il compito primario di questo coordinatore è salvaguardare il rigore scientifico del dato durante l'intero ciclo di vita dell'elaborazione. Per evitare troncamenti e perdita di informazioni, i tensori vengono forzatamente elaborati e mantenuti in virgola mobile ad alta precisione. A differenza dei comuni software di fotoritocco, Kometra [3] esporta i file finali preservando l'escursione dinamica completa, inclusi i fondamentali valori derivativi negativi, e aggiorna contestualmente la *keyword* `BITPIX` nell'header FITS. Infine, per garantire la totale riproducibilità scientifica dell'esperimento da parte di terzi, ogni parametro operativo impiegato (raggi, angoli, soglie calcolate) viene intercettato dal coordinatore e impresso nell'header del file di output sotto forma di record storico (`HISTORY`).

6.7 Interfaccia Utente: Applicazione Filtri

Per quanto riguarda l'interfaccia utente, l'accesso agli strumenti appena descritti è organizzato in modo logico all'interno del menu principale del software. Sotto il menu *Modifica* → *Analisi Morfologica*, l'utente può trovare tre categorie operative distinte, che riflettono le famiglie matematiche analizzate nelle sezioni precedenti: *Modelli Radiali*, *Estrazioni Strutture* e *Contrasto Locale*. Un punto importante riguarda il fatto che nessuna delle operazioni confermate tramite queste interfacce va mai a sovrascrivere i dati originali. Al termine dell'esecuzione di un *batch* di filtraggio, il *Coordinator* genera automaticamente un **Nuovo Nodo** nell'albero gerarchico del software, ramificandolo dal nodo sorgente.

Selezionando una delle categorie di analisi morfologica, il software istanzia un modello di finestra unificato, progettato per massimizzare lo spazio dedicato all'ispezione visiva. L'interfaccia è divisa verticalmente in due macro-aree strutturali:

- **Pannello di Controllo (Sinistra):** La porzione sinistra della finestra è dedicata alla configurazione algoritmica. Tramite un menu a tendina, l'utente seleziona lo specifico algoritmo da impiegare (es. filtro di Larson-Sekanina o Tensore di Struttura). In risposta a questa selezione, il pannello sottostante genera dinamicamente i soli controlli necessari (slider, campi numerici, checkbox) per l'algoritmo scelto. Per mitigare la curva di apprendimento e assistere l'astronomo nella calibrazione, ogni parametro è dotato di *tooltip* contestuali: posizionando il cursore sopra un controllo, il sistema espone una concisa spiegazione tecnica del suo effetto sulla matrice dati.
- ***Viewport* Interattiva (Destra):** Il pannello destro ospita un'interfaccia di *rendering* che restituisce l'anteprima del fotogramma in esame. Tale area consente all'utente di variare dinamicamente il fattore di scala dell'immagine e di aggiustare le soglie di *stretching* dell'istogramma visivo mediante il semplice scorrimento della rotella del mouse.

Il flusso di lavoro (*workflow*) per l'applicazione dei filtri si divide in due step di validazione. Inizialmente, l'utente configura i parametri e invoca il comando *"Calcola Anteprima"*. Questa azione innesca il motore matematico esclusivamente sull'immagine correntemente caricata nella *viewport*, permettendo di valutare l'efficacia del filtro e la presenza di eventuali artefatti in pochi millisecondi. Una volta identificato il *setup* ottimale, il comando *"Applica"* avvia l'elaborazione asincrona dell'intera sequenza di immagini contenuta nel nodo sorgente.

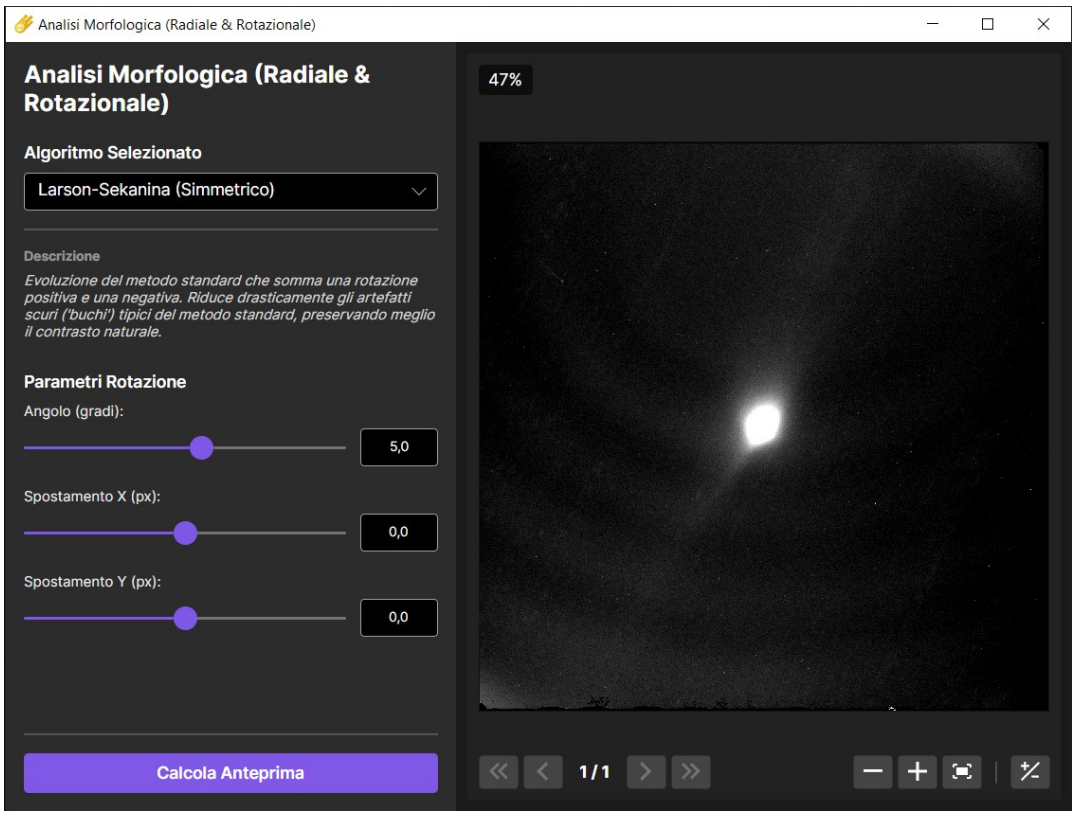

Figura 6.13. Interfaccia grafica unificata dedicata all'applicazione dei filtri morfologici. Nel pannello di sinistra sono raggruppati i controlli per la selezione dell'algoritmo e i relativi cursori di parametrizzazione. Sulla destra, l'area di visualizzazione restituisce l'anteprima del target cometario sottoposto a elaborazione.

Capitolo 7

Strumenti accessori per l'elaborazione visiva

L'indagine morfologica e dinamica delle comete, sebbene fortemente potenziata dall'impiego di operatori matematici e filtri spaziali complessi, non può prescindere da una fase di indagine visiva diretta e interattiva. Durante i frequenti confronti gli astrofili collaboratori, è emersa con chiarezza la necessità di affiancare ai motori di filtraggio automatizzati (discussi nel capitolo precedente) una suite di strumenti operativi classici, in modo da avere il pieno controllo manuale e decisionale sui dati.

I filtri discussi nel capitolo precedente aiutano nell'estrazione di micro-dettagli ad alta frequenza isolandoli dal gradiente della chioma, ma la loro natura prettamente algoritmica li rende meno adatti per valutazioni esplorative rapide, misurazioni differenziali dirette o per l'apprezzamento macroscopico dell'evoluzione temporale del corpo celeste. Per sopperire a queste necessità tipiche del flusso di lavoro astrofisico quotidiano, il software Kometra [3] è stato dotato di funzionalità accessorie progettate anche per il confronto tra immagini diverse.

Il presente capitolo illustrerà brevemente questi strumenti aggiuntivi, i quali si aggiungono alle altre funzionalità offerte dall'applicativo rendendolo un ambiente di lavoro completo. Verranno analizzate nel dettaglio tre funzionalità cardine: la possibilità di effettuare operazioni algebriche mirate tra diverse acquisizioni; l'ispezione temporale tramite la tecnica del *blinking*, fondamentale per cogliere le variazioni dinamiche tra fotogrammi successivi; e infine la *Posterizzazione*, una tecnica di quantizzazione radiometrica impiegata per la mappatura visiva delle isofote. Come per i capitoli precedenti, nei paragrafi conclusivi verranno inoltre descritte le scelte architetturali di backend e le logiche di interfaccia utente sviluppate per garantire l'esecuzione e il riscontro in tempo reale di tali operazioni.

7.1 Pixel Math: Operazioni Algebriche Interattive

Uno strumento fondamentale per l'analisi visiva è costituito dalla cosiddetta *Pixel Math*. A differenza dei filtri differenziali e spaziali discussi in precedenza, i quali eseguono trasformazioni complesse basate sul vicinato o sulle frequenze di un singolo fotogramma, la Pixel Math consente di eseguire operazioni aritmetiche dirette e lineari tra le matrici di pixel appartenenti a due nodi distinti. Gli operatori implementati all'interno del software comprendono le quattro operazioni fondamentali: addizione, sottrazione, moltiplicazione e divisione.

A livello ingegneristico, una delle problematiche più comuni nell'incrociare dati astrofisici risiede nella discrepanza dimensionale delle immagini, spesso derivante da ritagli (*crop*) differenti o dall'utilizzo di sensori eterogenei durante campagne osservative congiunte. Per risolvere questa criticità in modo trasparente per l'utente, le operazioni algebriche di Kometra [3] integrano un sistema di auto-allineamento geometrico. Basandosi sul prerequisito che le immagini siano già state allineate e centrate sul nucleo cometario, l'algoritmo fa coincidere matematicamente i centri

esatti delle due matrici. Il tensore risultante dall'operazione erediterà strettamente le dimensioni dell'immagine più piccola tra le due. Questa logica di intersezione spaziale previene la generazione di artefatti ai bordi e ottimizza l'uso della memoria, scartando le aree periferiche non coperte da entrambi i fotogrammi.

L'integrazione di questa funzionalità nel paradigma a nodi dello spazio di lavoro risulta particolarmente strategica e flessibile. Il motore logico alla base della *Pixel Math* è progettato per riconoscere dinamicamente il tipo di contenitore selezionato sulla *Board* (Nodo Singolo o Nodo Multiplo) e adattare la distribuzione del calcolo di conseguenza, gestendo in modo intelligente tre casistiche operative distinte:

- **Interazione tra Nodi Singoli (Singolo ↔ Singolo):**
  Costituisce la modalità di utilizzo più frequente e intuitiva. L'operazione aritmetica avviene in un rapporto uno a uno tra le due matrici selezionate.
- **Interazione Ibrida (Nodo Multiplo ↔ Nodo Singolo):**
  Quando l'utente seleziona una sequenza temporale (Nodo Multiplo) e un'immagine isolata (Nodo Singolo), l'algoritmo applica il fotogramma singolo come operatore costante per l'intero array del nodo multiplo. Questa casistica risulta vitale per abbattere i tempi operativi in scenari di calibrazione globale: permette, ad esempio, di dividere un'intera sequenza temporale di decine di fotogrammi per un singolo *Master Flat* di calibrazione, oppure di sottrarre in modalità batch un singolo modello sintetico della chioma da un intero *time-lapse* cometario, generando istantaneamente una nuova sequenza morfologicamente "pulita".
- **Interazione tra Nodi Multipli (Multiplo ↔ Multiplo):**
  L'applicativo consente infine di operare algebricamente tra due intere sequenze di immagini. Per garantire la coerenza dei dati, il motore impone un vincolo strutturale: i due nodi multipli devono possedere lo stesso numero di immagini. Soddisfatto questo requisito, l'operazione viene eseguita appaiando le immagini che condividono il medesimo indice interno.

I risultati vengono consolidati nella memoria volatile del sistema, generando un nuovo nodo (singolo o multiplo a seconda della casistica) che viene connesso visivamente ai suoi progenitori sulla *Board*, mantenendo inalterata la tracciabilità delle operazioni.

7.2 Ispezione Temporale e Tecnica del Blinking

L'ispezione visiva di sequenze temporali rappresenta un concetto fondamentale nell'indagine astronomica. In questo contesto, la tecnica del *blinking* (letteralmente "lampeggio" o alternanza rapida) consiste nel visualizzare in rapida successione a schermo due o più fotogrammi del medesimo soggetto acquisiti in istanti temporali differenti.

L'occhio umano, supportato dal fenomeno cognitivo della persistenza retinica, risulta straordinariamente sensibile ai cambiamenti dinamici che si verificano tra un'immagine e la successiva. Questa metodologia operativa permette ai ricercatori di cogliere in modo istantaneo informazioni cruciali: valutare visivamente il tasso di espansione e l'evoluzione morfologica dei getti di polvere, scovare con immediatezza eventuali artefatti transitori, nonché confrontare immagini ottenute da elaborazioni alternative.

In fase di progettazione dell'interfaccia di Kometra [3], anziché sviluppare un modulo di riproduzione separato o una finestra dedicata a questa tecnica, si è deciso di sfruttare l'elasticità strutturale offerta dall'architettura dello spazio di lavoro, utilizzando le capacità del nodo multiplo (`MultipleImagesNodeModel`).

L'applicativo mette infatti a disposizione dell'utente due comandi contestuali di gestione strutturale, progettati per garantire la massima fluidità dell'ambiente di lavoro:

- **Unione (Join):** Permette di selezionare due o più nodi singoli presenti sulla *Board* e di accorparli dinamicamente in un unico nodo multiplo.

- **Separazione (Split):** Esegue l'operazione inversa, permettendo di "esplodere" un nodo multiplo, estraendone i fotogrammi costituenti e separandoli in una griglia di nodi singoli indipendenti.

È proprio attraverso la sinergia tra la funzionalità di *Join* e il nodo multiplo che si concretizza l'esecuzione del blinking. Per avviare l'analisi, l'operatore seleziona i nodi singoli contenenti la sequenza temporale di interesse (preventivamente allineata sul nucleo della cometa) e ne comanda l'unione. Una volta che le immagini convergono nell'unico contenitore multiplo, premendo semplicemente la barra spaziatrice, l'utente innesca l'animazione automatica della sequenza. Il motore dell'interfaccia inizia ad alternare ciclicamente la renderizzazione a schermo delle immagini interne al nodo con una frequenza preimpostata, generando un blinking fluido e reattivo direttamente all'interno della cornice del nodo sulla *Board*.

Questa implementazione garantisce un flusso di lavoro ininterrotto: qualora il blinking evidenziasse un fotogramma degradato o la necessità di analizzare un'anomalia su una singola posa, l'operatore può interrompere l'animazione, invocare il comando di separazione (*Split*) per riottenere le immagini su nodi singoli e procedere immediatamente all'esclusione del file o all'applicazione di un filtro morfologico differenziale, senza mai dover abbandonare l'area di lavoro principale.

7.3 Mappatura Morfologica tramite Quantizzazione (Posterizzazione)

Accanto all'ispezione temporale e alle operazioni algebriche descritte in precedenza, Kometra [3] offre un ulteriore strumento di analisi visiva diretta basato sulla quantizzazione radiometrica del segnale: la *Posterizzazione*.

In astrofotografia, e in particolare nello studio cometario, la posterizzazione non costituisce un mero effetto estetico, bensì un rigoroso strumento di indagine morfologica. Riducendo l'ampia gamma dinamica originaria in virgola mobile a un numero discreto di livelli tonali, l'algoritmo produce una mappatura diretta delle *isofote* (regioni a isointensità fotometrica). Così facendo, questa quantizzazione visiva consente all'osservatore di isolare le asimmetrie strutturali della chioma, agevolando la percezione immediata delle deviazioni dei gradienti rispetto a una distribuzione sferica isotropa.

Dal punto di vista architetturale, l'elaborazione si articola in tre fasi matematiche distinte, ideate per operare rigorosamente in virgola mobile al fine di scongiurare errori di arrotondamento:

1. **Normalizzazione Lineare:** L'algoritmo riceve in ingresso i valori di taglio inferiore (*Black Point*) e superiore (*White Point*) del segnale utile. Tramite una trasformazione affine, i dati FITS grezzi vengono mappati esattamente nell'intervallo continuo $[0.0, 1.0]$. Un meccanismo di *clipping* vincola i valori esterni (*outliers*) ai limiti di questo range.

2. **Compressione Dinamica (Curva di Trasferimento):** Essendo il nucleo cometario intrinsecamente sovraesposto rispetto alla debolissima chioma esterna, una quantizzazione puramente lineare allocherebbe la quasi totalità dei livelli disponibili alle regioni centrali. Per compensare questo squilibrio fotometrico, il software consente l'applicazione opzionale di funzioni di trasferimento non lineari prima della discretizzazione. È supportata la trasformazione a *Radice Quadrata* per comprimere dolcemente le alte luci, e la trasformazione *Logaritmica* per esaltare in modo aggressivo i gradienti più deboli del fondo cielo, ripristinando al termine l'intervallo normalizzato.

3. **Quantizzazione a Livelli:** La fase di quantizzazione vera e propria avviene moltiplicando il segnale normalizzato per il numero di livelli desiderati e applicando una funzione di troncamento all'intero inferiore tramite *casting* esplicito a matrice intera a 32 bit. Il tensore risultante viene successivamente riconvertito in virgola mobile e ri-scalato nell'intervallo canonico $[0, 255]$ per l'esportazione finale e la renderizzazione a schermo.

È fondamentale sottolineare la natura *fortemente distruttiva* di questa operazione. A differenza dei modelli radiali o differenziali che mirano a isolare le frequenze preservando l'integrità del dato fisico sottostante, la riduzione letterale del numero di livelli tonali effettivi abbatte drasticamente la risoluzione fotometrica originaria del sensore. Le micro-variazioni e i deboli gradienti sub-decimali vengono compressi e fusi in modo irreversibile all'interno della medesima banda quantizzata. Di conseguenza, l'immagine posterizzata deve essere considerata unicamente come un ausilio qualitativo per l'ispezione morfologica macroscopica, risultando totalmente inidonea per successive misurazioni astrometriche o fotometriche di precisione.

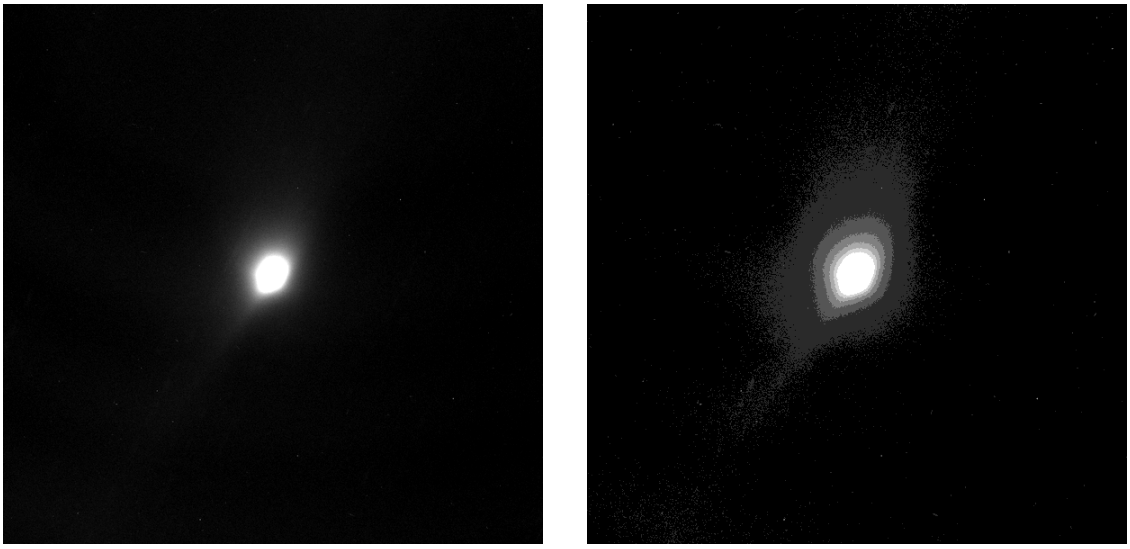

Figura 7.1. Estrazione visiva delle isofote tramite quantizzazione non lineare. A sinistra, l'immagine originale della cometa 67P in formato lineare ad alta dinamica. A destra, il risultato dell'algoritmo di posterizzazione applicato con curva di trasferimento logaritmica.

7.4 Architettura Software: Strumenti Accessori

In stretta continuità con le scelte architetturali delineate nei capitoli precedenti, l'implementazione degli strumenti accessori rispetta la rigorosa dicotomia tra motori di calcolo puro (*Engines*) e moduli di orchestrazione del flusso dati (*Coordinators*).

Scendendo al livello degli *Engines*, la logica matematica della *Pixel Math* è incapsulata all'interno dell'`ArithmeticEngine`. Poiché questo motore accetta e restituisce esclusivamente tensori (`OpenCvSharp.Mat`), la gestione delle discrepanze geometriche tra le immagini in ingresso viene risolta appoggiandosi al motore `GeometricEngine`. Quest'ultimo si occupa di calcolare la centratura esatta del secondo operando rispetto al *canvas* del primo, garantendo un allineamento sub-pixel perfetto. Un aspetto computazionale critico gestito a questo livello è la preservazione della precisione: il motore interroga la profondità in bit delle matrici d'ingresso e, qualora rilevi la presenza di dati in doppia precisione a 64 bit, promuove automaticamente l'intero *workspace* temporaneo e il risultato finale al formato `Double`. Parallelamente, per prevenire instabilità a *runtime* derivanti da aree non valide o da divisioni per zero, la routine impiega un massiccio sistema di mascheramento logico (`Cv2.Compare` e `Cv2.BitwiseAnd`). Le aree in cui i dati mancano o le divisioni non sono matematicamente calcolabili vengono isolate a monte e forzatamente re-impostate a `NaN` sul tensore di output, garantendo la totale sicurezza numerica dell'applicativo.

Invece, al livello superiore, l'orchestrazione della logica di dominio e l'interazione con il *file system* FITS sono ripartite su tre classi *Dispatcher* centrali: il `NodeStructureCoordinator`, l'`ArithmeticCoordinator` e il `PosterizationCoordinator`.

Il `NodeStructureCoordinator` presiede alla complessa operazione di *Join* per il *blinking*. Anziché effettuare banali ritagli o interpolazioni distruttive su immagini di dimensioni eterogenee, il

coordinatore interroga in modo asincrono tutti gli *Header* della sequenza selezionata estraendo le chiavi `NAXIS1` e `NAXIS2`. Da questi metadati ricava la *Bounding Box* massima, generando un *canvas* universale sul quale viene incollato ogni fotogramma, consentendo un'animazione temporale perfettamente stabile. L'`ArithmeticCoordinator` gestisce invece l'intera esecuzione delle operazioni aritmetiche tra i nodi (la *Pixel Math*). Oltre a orchestrare il calcolo algebrico vero e proprio, esso risolve le differenze nella quantità di immagini contenute nei nodi selezionati. Il coordinatore valuta i dati in ingresso per determinare come accoppiarli: se i due nodi contengono sequenze della stessa lunghezza, l'operazione avviene per coppie ordinate (associando la prima immagine del primo nodo con la prima del secondo, la seconda con la seconda, e così via); se invece si opera tra un nodo multiplo e uno singolo, l'immagine singola viene mantenuta fissa e applicata matematicamente a tutti i fotogrammi della sequenza. Questa seconda casistica risulta fondamentale, ad esempio, per sottrarre in un solo passaggio un singolo *Master Frame* di calibrazione da un intero *dataset*.

Un obbligo architetturale comune imposto a questi due coordinatori riguarda la sanitizzazione dei metadati. Dal momento che le manipolazioni convertono invariabilmente i dati grezzi in matrici in virgola mobile, i coordinatori sfruttano il`FitsMetadataService` per resettare forzatamente i parametri di scala (`BSCALE = 1.0` e `BZERO = 0.0`). Questa operazione è vitale per impedire che i software di lettura terzi riapplichino offset pensati per vecchi interi a 16 bit su matrici *floating-point*. Oltre a ciò, il coordinatore ricalcola i valori fotometrici di picco (`DATAMIN` e `DATAMAX`) e inietta nel record `HISTORY` tutti i riferimenti dei file operandi, assicurando la totale riproducibilità scientifica dell'operazione matematica eseguita.

Infine, la mappatura morfologica delle isofote tramite quantizzazione radiometrica è orchestrata dal `PosterizationCoordinator`. Per garantire che l'interfaccia utente aggiorni l'immagine istantaneamente mentre l'operatore sposta i cursori, questo modulo divide il processo in due fasi distinte, separando l'anteprima visiva dal calcolo scientifico reale. Durante la regolazione dei parametri, il coordinatore genera un'anteprima rapida operando su una versione alleggerita dell'immagine a 8 bit, ottimizzata unicamente per l'aggiornamento del monitor senza sovraccaricare il processore. Solo al momento della conferma da parte dell'utente, l'elaborazione definitiva sui pesanti dati FITS originali viene avviata in background tramite l'`IBatchProcessingService`.

Durante l'elaborazione dell'intera sequenza temporale interviene un'ulteriore funzionalità chiave: l'adattamento dinamico (*Auto-Adapt*). Processare una sequenza di immagini presenta una criticità pratica: la luminosità della cometa o le condizioni di esposizione possono variare sensibilmente da un fotogramma all'altro. Se il software imponesse limiti di nero e di bianco rigidi e identici per tutta la sequenza, i fotogrammi leggermente più chiari o più scuri rispetto al primo risulterebbero illeggibili. Abilitando l'*Auto-Adapt*, il coordinatore supera questo limite analizzando il profilo statistico (Media e Deviazione Standard) di ogni singola immagine tramite l'`IImagePresentationService`. Basandosi su questi dati, l'algoritmo ricalcola in automatico le soglie di taglio ottimali fotogramma per fotogramma, garantendo che le isofote estratte risultino sempre coerenti e ben contrastate nonostante le naturali fluttuazioni luminose dell'acquisizione.

7.5 Interfaccia Utente: Analisi Temporale, Algebrica e Posterizzazione

L'efficacia degli strumenti esplorativi accessori risiede nella loro immediata accessibilità e nella fluidità con cui l'operatore può richiamarli direttamente dallo spazio di lavoro principale. L'interfaccia utente di questi strumenti è stata pertanto modellata per minimizzare l'attrito cognitivo, affidando le manipolazioni strutturali e matematiche a selezioni visive intuitive e a comandi di menu contestuali.

7.5.1 Join, Split e Ispezione Temporale (Blinking)

Per supportare la rapida ispezione temporale, l'interfaccia permette di riconfigurare dinamicamente i contenitori delle immagini. Selezionando due o più nodi singoli sulla *Board*, l'utente può

accorparli in una singola sequenza temporale navigando nel menu principale alla voce *Modifica* → *Unisci nodi*. L'operazione inversa, utile per isolare fotogrammi difettosi o per analizzare singole pose, è altrettanto immediata tramite il comando *Modifica* → *Dividi nodo*.

Una volta ottenuto o caricato un nodo multiplo, l'analisi dinamica (il *blinking*) non richiede l'apertura di finestre secondarie. È sufficiente selezionare il nodo contenitore e premere la barra spaziatrice sulla tastiera per innescare immediatamente l'alternanza ciclica dei fotogrammi a schermo. In alternativa, la medesima animazione può essere avviata e fermata tramite il comando *Visualizza* → *Avvia animazione*.

7.5.2 Interazione Algebrica (Pixel Math)

L'esecuzione delle operazioni aritmetiche richiede un controllo logico più rigoroso, in quanto l'interfaccia deve garantire la corretta interpretazione matematica dell'azione. Le voci del menu relative a questa funzionalità, raggruppate sotto *Modifica* → *Operazioni aritmetiche*, vengono abilitate esclusivamente quando l'operatore seleziona esattamente due nodi validi sulla *Board*.

Poiché per operatori come la sottrazione o la divisione l'ordine dei fattori non è commutativo ed è matematicamente vincolante, è indispensabile che l'utente sappia in ogni momento quale immagine viene considerata come primo operando e quale come secondo. Per risolvere questa ambiguità in modo elegante, l'interfaccia utente adotta un ausilio visivo contestuale: al momento della selezione multipla, accanto alle cornici dei nodi scelti compaiono dei *badge* grafici contenenti le lettere "A" e "B" (come visibile in Figura 7.2). Questo marcatore esplicita quale nodo fungerà da base (A) e quale fungerà da modificatore (B), garantendo la totale prevedibilità del risultato calcolato.

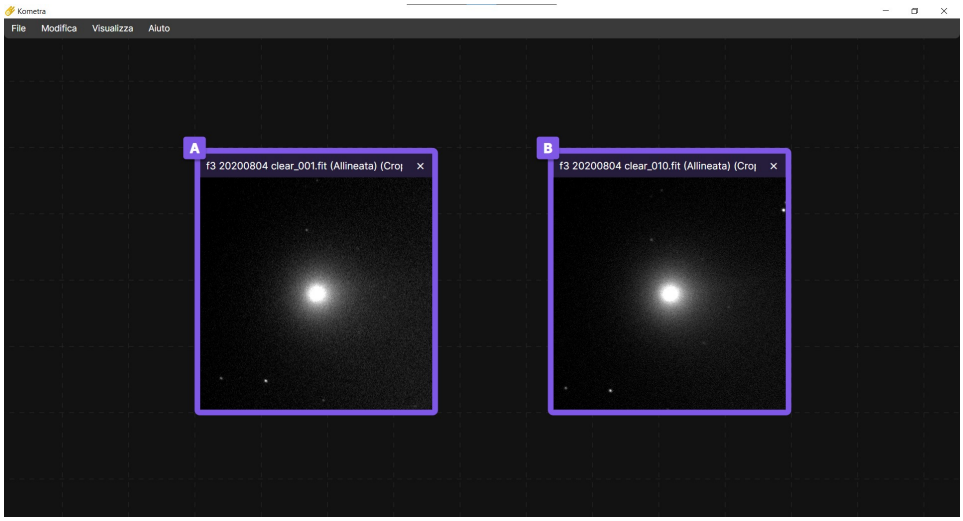

Figura 7.2. Interfaccia di selezione per le operazioni aritmetiche (Pixel Math). Selezionando due nodi sulla Board, l'interfaccia genera dinamicamente le etichette visive "A" e "B". Questo ausilio grafico è essenziale per definire l'ordine degli operandi prima di invocare operazioni non commutative come la sottrazione o la divisione.

7.5.3 La Finestra di Quantizzazione e Posterizzazione

La finestra dedicata alla posterizzazione eredita il medesimo impianto ergonomico della maggior parte degli strumenti dell'applicazione (Pannello di controllo a sinistra, *Viewport* a destra), ma introduce un paradigma di interazione in *Real-Time*.

Essendo la quantizzazione un'operazione visivamente molto sensibile ai minimi spostamenti di soglia, il pannello espone tre controlli fondamentali: il *Numero di Livelli* di discretizzazione desiderati (isofote), un selettore per la *Curva di Distribuzione* (Lineare, Logaritmica o Radice

Quadrata) e i cursori per la definizione esatta delle *Soglie di Taglio* (Punto di Nero e Punto di Bianco).

A differenza dei complessi filtri di analisi morfologica, ogni interazione dell'utente con questi controlli innesca un ricalcolo visivo immediato (*live update*) della mappa delle isofote all'interno della *viewport*. Questa caratteristica offre un livello di responsività fondamentale per il *tuning* fine dei gradienti sul singolo fotogramma. Solo a valle di questa calibrazione visiva, la conferma finale innesca il processo di quantizzazione massiva e asincrona sull'intero *dataset* sottostante.

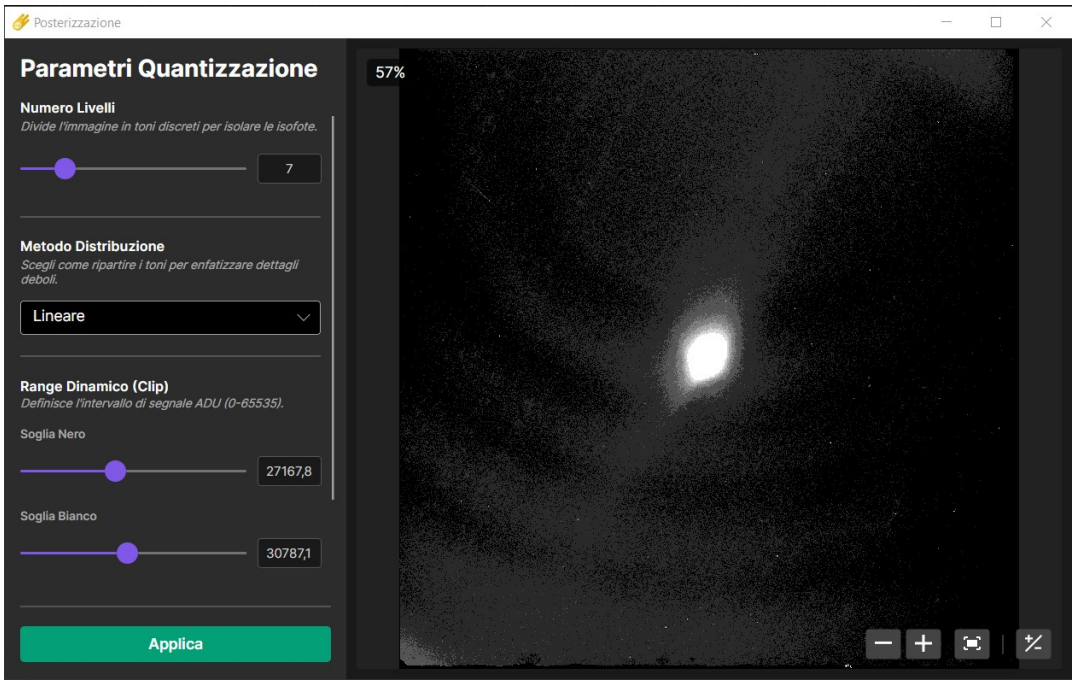

Figura 7.3. Interfaccia di posterizzazione adattiva dinamica: a sinistra i controlli per la parametrizzazione dei livelli (isofote), la curva tonale e i limiti di intensità (ADU); a destra l'anteprima modificata in tempo reale che illustra l'immagine cometaria discretizzata in bande tonali distinte.

Capitolo 8

Conclusione

L'obiettivo primario di questo lavoro di tesi è stato la progettazione e lo sviluppo di Kometra [3], un sistema software integrato per l'elaborazione astrometrica e l'analisi morfologica avanzata di immagini cometarie. Partendo dall'analisi delle criticità che affliggono i tradizionali flussi di lavoro astrofotografici (frammentarietà degli strumenti, obsolescenza delle interfacce e inefficienza nella gestione di dati scientifici dinamici) si è delineata la necessità di un'architettura software moderna, capace di coniugare rigore matematico e usabilità operativa.

Il risultato di questo percorso ingegneristico è un applicativo multipiattaforma, sviluppato in C# su *framework* .NET con interfaccia Avalonia UI, in grado di coprire l'intero ciclo di vita del dato astronomico. La stretta collaborazione con astrofili ha permesso di validare iterativamente i requisiti di dominio, garantendo che le scelte algoritmiche e architetturali rispondessero a problematiche scientifiche reali e misurabili.

Dal punto di vista dell'Ingegneria del Software, il progetto ha dimostrato come l'applicazione di *design pattern* e la rigida *separation of concerns* tra motori di calcolo e livelli di orchestrazione sia essenziale per gestire la complessità di un dominio altamente specializzato. I risultati tecnici più significativi raggiunti in questa trattazione possono essere riassunti in quattro macro-aree:

- **Gestione Dati e Ottimizzazione della Memoria:** L'implementazione da zero di un *parser* proprietario ha svincolato il software da dipendenze esterne obsolete, dotandolo della flessibilità necessaria per elaborare la quasi totalità delle tipologie e varianti di file FITS. Il sistema è infatti in grado di adattarsi in modo trasparente all'elevata variabilità strutturale che caratterizza questo standard (Multi-Extension FITS, tipizzazioni numeriche eterogenee e calibrazioni di scala). A questo si aggiunge lo sviluppo dei motori di codifica *bit-exact* per la *Tiled Image Compression* (algoritmi Rice e GZIP), che ha garantito una piena interoperabilità scientifica. A completamento della fase di pre-elaborazione, è stata inoltre integrata una gestione rigorosa della calibrazione strumentale (Bias, Dark, Flat), assicurando la corretta normalizzazione del dato prima delle analisi morfologiche. Parallelamente, le criticità legate all'elaborazione di tensori massivi in virgola mobile sono state risolte tramite un'oculata gestione della memoria e il *throttling* concorrenziale, prevenendo la saturazione del *Garbage Collector* e ottimizzando i tempi di esecuzione in ambiente *multithreading*.
- **Identificazione sub-pixel del Nucleo e Tracking Dinamico:** Un aspetto critico affrontato durante lo sviluppo ha riguardato il tracciamento di oggetti intrinsecamente diffusi e asimmetrici. Attraverso una fase di *benchmarking* comparativo condotta su svariate famiglie di algoritmi, è stato selezionato e ottimizzato il modello matematico che ha dimostrato la maggiore stabilità e ripetibilità sui dati reali. Questo approccio ha consentito di stimare le coordinate dell'optocentro con precisione sub-pixel, garantendo l'accuratezza necessaria per la corretta registrazione geometrica dei fotogrammi in vista delle successive operazioni di *stacking*. A supporto di questa procedura, l'integrazione con i database effemeridali del JPL e i risolutori astrometrici (ASTAP) ha permesso di automatizzare in modo affidabile la fase di calibrazione spaziale.

- **Segmentazione e Inpainting (Starless):** La *pipeline* ideata per la rimozione delle stelle di campo ha dimostrato come la combinazione di morfologia matematica e tecniche di *inpainting* stocastico possa preservare l'integrità fotometrica della chioma cometaria, restituendo risultati qualitativamente eccellenti e pienamente competitivi rispetto agli applicativi attualmente presenti sul mercato. Questo approccio ha così eliminato la necessità di ricorrere a software di terze parti, centralizzando l'intero flusso di lavoro all'interno di un unico ambiente integrato.
- **Studio Morfologico e Ingegnerizzazione dei Filtri:** La trasposizione in codice di complessi modelli matematici radiali, differenziali e tensoriali, mutuati da diversi ambiti scientifici, ha permesso di condurre uno studio comparativo approfondito. L'analisi ha dimostrato come alcuni approcci algoritmici trasversali risultino inefficaci o inadatti allo specifico studio morfologico delle polveri cometarie. Al contrario, i filtri astronomici più classici si sono confermati lo strumento migliore: essi sono stati profondamente reingegnerizzati con estremo rigore matematico e ottimizzati architetturalmente. L'intera *suite* analitica è stata resa fruibile attraverso un'interfaccia utente proattiva e un paradigma non distruttivo che, grazie ad anteprime in tempo reale, abbatte la curva di apprendimento. In quest'ottica, l'introduzione di strumenti di manipolazione diretta come la Pixel Math permette l'esecuzione di operazioni algebriche tra i nodi della Board, facilitando analisi differenziali complesse senza alcun rischio di alterazione dei dati originali.

In un'ottica di massima trasparenza scientifica e per favorire la cooperazione internazionale nel settore, Kometra è stato distribuito come software *open-source* sotto licenza GNU GPL v3.0. Il codice sorgente è quindi pubblicamente accessibile tramite il repository GitHub all'indirizzo `https://github.com/Genofabio/Kometra` [3]. Inoltre, per supportare la collaborazione tra team di ricerca eterogenei, il software integra nativamente il supporto multilingua (italiano e inglese), con un'architettura basata su file di risorse già predisposta per l'aggiunta semplificata di ulteriori idiomi.

In sintesi, Kometra [3] non si configura unicamente come un aggregatore di algoritmi matematici, ma come un'infrastruttura ingegneristica completa, dimostrando che è possibile modernizzare lo studio morfologico delle comete massimizzando l'estrazione dell'informazione scientifica pur mantenendo un'esperienza utente fluida e sicura, garantendo al contempo un'esportazione versatile sia verso formati scientifici che divulgativi.

Sebbene l'architettura attuale soddisfi pienamente i requisiti delineati in fase di analisi, la natura modulare del codice apre la strada a molteplici traiettorie di sviluppo. Sotto il profilo computazionale, l'implementazione dell'accelerazione hardware tramite Graphics Processing Unit (GPU) consentirebbe di parallelizzare i carichi di lavoro più onerosi, abbattendo ulteriormente i tempi di esecuzione dei filtri spaziali. Parallelamente, l'integrazione di architetture di *Deep Learning* (reti neurali convoluzionali) per la rimozione del campo stellare permetterebbe di superare i limiti fisiologici della morfologia matematica, garantendo una segmentazione semantica ancora più accurata. Dal punto di vista prettamente astrofisico, le funzionalità del software potrebbero essere estese integrando un motore per la fotometria d'apertura rigorosa, capace di interrogare cataloghi stellari online per la calibrazione radiometrica. Infine, il software potrebbe essere esteso con un modulo dedicato alla simulazione tridimensionale della dinamica delle polveri. L'integrazione visiva tra i modelli cinematici (sindinamiche e sincrone) e i fotogrammi elaborati consentirebbe di datare con esattezza l'emissione dei getti e i fenomeni di frammentazione.

Bibliografia

[1] *Definition of the Flexible Image Transport System (FITS)*, ver. 4.0, NASA/GSFC, 2016. indirizzo: `https://fits.gsfc.nasa.gov/standard40/fits_standard40aa-le.pdf`.

[2] W. D. Pence, R. Seaman e R. L. White, «Tiled image convention for storing compressed images in FITS binary tables», *Publications of the Astronomical Society of the Pacific*, vol. 121, n. 879, p. 414, 2009.

[3] F. Genovese, *Kometra: An integrated, cross-platform software system for astrometric processing and advanced morphological analysis of cometary images*, ver. 1.0.0, 2026. indirizzo: `https://github.com/Genofabio/Kometra`.

[4] Avalonia UI, *Avalonia: A cross-platform UI framework for .NET*, 2024. indirizzo: `https://avaloniaui.net/` (visitato il giorno 04/03/2026).

[5] G. Bradski e A. Kaehler, *Learning OpenCV: Computer vision with the OpenCV library*. O'Reilly Media, Inc., 2008.

[6] Shimat, *OpenCvSharp: OpenCV wrapper for .NET*, 2024. indirizzo: `https://github.com/shimat/opencvsharp`.

[7] S. M. Larson e Z. Sekanina, «Coma morphology and dust-emission pattern of periodic Comet Halley. I-High-resolution images taken at Mount Wilson in 1910», *The Astronomical Journal*, vol. 89, n. 4, pp. 571–578, 1984.

[8] J. Smith, «WPF apps with the model-view-viewmodel design pattern», *MSDN magazine*, vol. 24, n. 2, pp. 72–82, 2009.

[9] E. Gamma, R. Helm, R. Johnson e J. Vlissides, *Design patterns: elements of reusable object-oriented software*. Pearson Education, 1994.

[10] M. Seemann e S. van Deursen, *Dependency Injection Principles, Practices, and Patterns*. Manning Publications, 2019.

[11] R. C. Martin, *Agile software development: principles, patterns, and practices*. Prentice Hall, 2003.

[12] P. Teuben, B. Thomas, L. Shaya et al., «Once FITS, Always FITS? Astronomical Infrastructure in Transition», *arXiv preprint arXiv:1809.09224*, 2018. indirizzo: `https://arxiv.org/abs/1809.09224`.

[13] D. C. Wells, E. W. Greisen e R. H. Harten, «FITS: A Flexible Image Transport System», *Astronomy and Astrophysics Supplement Series*, vol. 44, pp. 363–374, 1981.

[14] NASA/GSFC. «Registry of FITS Conventions». (), indirizzo: `https://fits.gsfc.nasa.gov/fits_registry.html` (visitato il giorno 10/02/2026).

[15] «FITS Primer», NASA Goddard Space Flight Center. (), indirizzo: `https://fits.gsfc.nasa.gov/fits_primer.html` (visitato il giorno 10/02/2026).

[16] «FITS Overview», HEASARC - NASA. (), indirizzo: `https://heasarc.gsfc.nasa.gov/docs/heasarc/fits_overview.html` (visitato il giorno 09/02/2026).

[17] R. H. Harten, P. Grosbol, E. W. Greisen e D. C. Wells, «The FITS tables extension», *Astronomy and Astrophysics Supplement Series*, vol. 73, pp. 365–372, 1988.

[18] W. D. Cotton, D. Tody e W. D. Pence, «Binary table extension to FITS», *Astronomy and Astrophysics Supplement Series*, vol. 113, pp. 159–166, 1995.

[19] R. L. White, P. Greenfield, W. Pence, D. Tody e R. Seaman, «Compression for the Virtual Observatory», *arXiv preprint arXiv:1201.1336*, 2012. indirizzo: https://arxiv.org/abs/1201.1336.

[20] D. J. Eicher, *COMETS!* Cambridge University Press, ott. 2013, Foreword by David H. Levy, ISBN: 9781107045255. DOI: 10.1017/CBO9781107045255. indirizzo: https://doi.org/10.1017/CBO9781107045255.

[21] NASA. «Kuiper Belt: Facts», National Aeronautics e Space Administration. (), indirizzo: https://science.nasa.gov/solar-system/kuiper-belt/facts/ (visitato il giorno 25/02/2026).

[22] NASA. «Oort Cloud: Facts», National Aeronautics e Space Administration. (), indirizzo: https://science.nasa.gov/solar-system/oort-cloud/facts/ (visitato il giorno 25/02/2026).

[23] K. J. Meech, «Setting the scene: what did we know before Rosetta?», *Philosophical Transactions of the Royal Society A*, vol. 375, p. 20 160 247, 2017. DOI: 10.1098/rsta.2016.0247. indirizzo: http://dx.doi.org/10.1098/rsta.2016.0247.

[24] K.-H. Glassmeier, H. Boehnhardt, D. Koschny, E. Kührt e I. Richter, «The ROSETTA Mission: Flying Towards the Origin of the Solar System», *Space Science Reviews*, vol. 128, n. 1-4, pp. 1–21, 2007. DOI: 10.1007/s11214-006-9140-8.

[25] European Space Agency, *Structure of a comet*, Accesso: 2026-03-13, 2023. indirizzo: https://www.esa.int/ESA_Multimedia/Images/2023/11/Structure_of_a_comet.

[26] M. G. G. T. Taylor, N. Altobelli, B. J. Buratti e M. Choukroun, «The Rosetta mission orbiter science overview: the comet phase», *Philosophical Transactions of the Royal Society A*, vol. 375, n. 2097, p. 20 160 262, lug. 2017. DOI: 10.1098/rsta.2016.0262. indirizzo: https://doi.org/10.1098/rsta.2016.0262.

[27] D. Prialnik e H. Sierks, «A mechanism for comet surface collapse as observed by Rosetta on 67P/Churyumov-Gerasimenko», *Monthly Notices of the Royal Astronomical Society*, vol. 469, n. Suppl_2, S217–S221, 2017.

[28] J.-Y. Li et al., «Photometric Properties of the Nucleus of Comet 103P/Hartley 2», *Icarus*, vol. 222, n. 2, pp. 559–570, 2013. DOI: 10.1016/j.icarus.2012.11.001.

[29] D. Prialnik e D. Jewitt, *Amorphous Ice in Comets: Evidence and Consequences*, Book chapter preprint, 2022. indirizzo: https://faculty.epss.ucla.edu/~jewitt/papers/2022/PJ22.pdf.

[30] K. Altwegg, H. Balsiger e S. A. Fuselier, «Cometary Chemistry and the Origin of Icy Solar System Bodies: The View After Rosetta», *arXiv preprint arXiv:1908.04046*, 2019. indirizzo: https://arxiv.org/abs/1908.04046.

[31] D. Jewitt, «From Comets to Asteroids: When Hairy Stars Go Bald», in *Proceedings of the 1994 Small Bodies meeting in Mariehamn, Finland*, Invited review lecture, 1996.

[32] G. D. Brin e D. A. Mendis, «Dust release and mantle development in comets», *The Astrophysical Journal*, vol. 229, pp. 402–408, apr. 1979. DOI: 10.1086/156954.

[33] M. Serra-Ricart, J. Licandro e M. R. Alarcon, «Pre-perihelion detection of a wobbling high-latitude jet in the interstellar comet 3I/ATLAS», *Astronomy & Astrophysics (submitted)*, 2025, Osservazioni condotte con il Two-meter Twin Telescope (TTT) presso l'Osservatorio del Teide.

[34] M. S. P. Kelley, S. Protopapa, D. Bodewits et al., «A Large Outburst, Coma Asymmetries, and the Color of Comet 243P/NEAT», *The Planetary Science Journal (Accepted)*, 2025, Studio sulle asimmetrie della chioma e l'impatto degli outburst sulla fotometria e l'optocentro.

[35] V. Rosenbush, V. Kleshchonok, O. Ivanova et al., «A comprehensive study of comet 67P/Churyumov-Gerasimenko in the 2021/2022 apparition. I. Photometry, spectroscopy, morphology», *Icarus*, vol. 444, p. 116 799, 2026, ISSN: 0019-1035. DOI: 10.1016/j.icarus.2025.116799.

[36] Gaia Collaboration, A. Vallenari, A. G. A. Brown, T. Prusti, J. H. J. de Bruijne et al., «Gaia Data Release 3: Summary of the content and survey properties», *Astronomy & Astrophysics*, vol. 674, A1, 2023. DOI: 10.1051/0004-6361/202243940. indirizzo: https://doi.org/10.1051/0004-6361/202243940.

[37] H. Kleijn, *ASTAP star pattern recognition algorithm and astrometric (plate) solving*, https://www.hnsky.org/astap_astrometric_solving.htm, Accesso: 26 Febbraio 2026, 2025.

[38] E. W. Greisen e M. R. Calabretta, «Representations of world coordinates in FITS», *Astronomy & Astrophysics*, vol. 395, pp. 1061–1075, 2002, Documento caricato: 0207407v2.pdf.

[39] M. R. Calabretta e E. W. Greisen, «Representations of celestial coordinates in FITS», *Astronomy & Astrophysics*, vol. 395, pp. 1077–1122, 2002, Documento caricato: aah3860.pdf.

[40] D. L. Shupe, M. Moshir, J. Li, D. Makovoz, R. Narron e R. N. Hook, «The SIP convention for representing distortion in FITS image headers», in *Astronomical Data Analysis Software and Systems XIV*, vol. 347, 2005, p. 491.

[41] H. Kleijn, *ASTAP Command Line Interface and Integration*, https://www.hnsky.org/astap.htm#command_line, Accesso: 26 Febbraio 2026, 2025.

[42] J. D. Giorgini, D. K. Yeomans, A. B. Chamberlin, P. W. Chodas et al., «JPL's On-Line Solar System Data Service», *Bulletin of the American Astronomical Society*, vol. 28, p. 1158, 1996.

[43] R. S. Park, W. M. Folkner, J. G. Williams e D. H. Boggs, «The JPL Planetary and Lunar Ephemerides DE440 and DE441», *The Astronomical Journal*, vol. 161, n. 3, p. 105, 2021. DOI: 10.3847/1538-3881/abd411.

[44] D. Farnocchia, S. R. Chesley, M. Micheli et al., «High precision comet trajectory estimates: The Mars flyby of C/2013 A1 (Siding Spring)», *Icarus*, vol. 266, pp. 279–287, 2016. DOI: 10.1016/j.icarus.2015.10.035.

[45] B. G. Marsden, Z. Sekanina e D. K. Yeomans, «Comets and nongravitational forces. V», *The Astronomical Journal*, vol. 78, pp. 211–225, 1973.

[46] S. B. Howell, *Handbook of CCD Astronomy*, 2ª ed. Cambridge, UK: Cambridge University Press, 2006, ISBN: 978-0-521-85215-9.

[47] M. Nicolini, «Due algoritmi per l'elaborazione morfologica delle chiome cometarie», *CARA Project (Cometary Archive for Amateur Astronomer)*, 2015, Rev. Mauro Facchini.

[48] T. Bonev e K. Jockers, «Spatial distribution of the dust color in comet C/LINEAR (2000 WM1)», in *Proceedings of Asteroids, Comets, Meteors - ACM 2002*, B. Warmbein, cur., vol. 500, Noordwijk, Netherlands: ESA Publications Division, 2002, pp. 587–591.

[49] N. H. Samarasinha, M. P. Martin e S. M. Larson, *Cometary Coma Image Enhancement Facility*, http://cie.psi.edu, 2013.

[50] S. M. Pizer, E. P. Amburn, J. D. Austin et al., «Adaptive histogram equalization and its variations», *Computer vision, graphics, and image processing*, vol. 39, n. 3, pp. 355–368, 1987.

[51] A. F. Frangi, W. J. Niessen, K. L. Vincken e M. A. Viergever, «Multiscale vessel enhancement filtering», in *Medical Image Computing and Computer-Assisted Intervention—MICCAI'98*, Springer, 1998, pp. 130–137.

[52] J. Bigün e G. H. Granlund, «Optimal orientation detection of linear symmetry», *First International Conference on Computer Vision, London*, pp. 433–438, 1987.