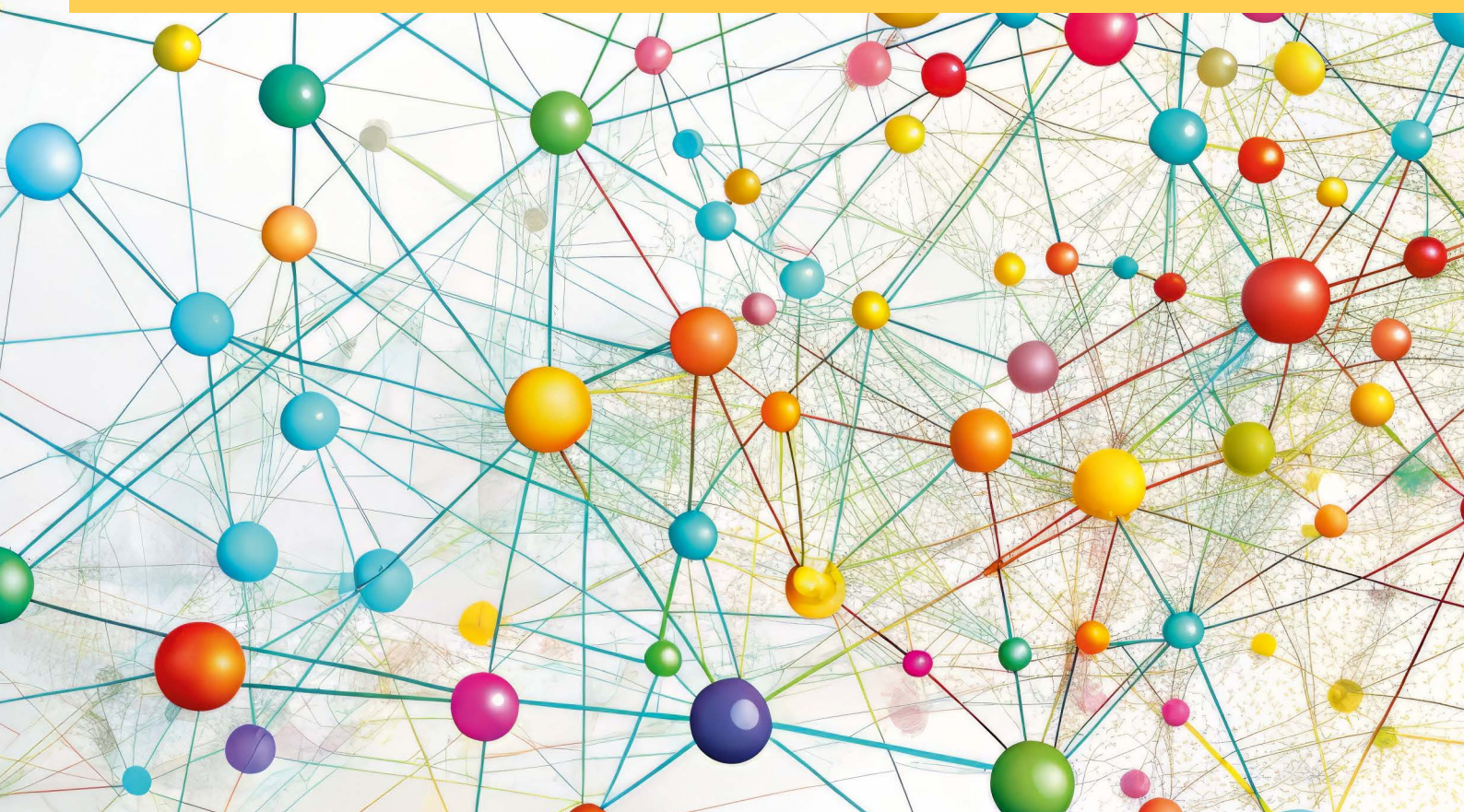




RAPPORTI TECNICI INGV

PyGraph Scanner
uno strumento interattivo per
digitalizzare serie storiche



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

513

**Direttore Responsabile**

Daniela VERSACE

Editor in Chief

Milena MORETTI

Editorial Board

Laura ALFONSI

Sofia BARANELLO

Carla BARNABA

Christian BIGNAMI

Simona CARANNANTE

Maria Grazia CIACCIO

Luca COCCHI

Luigi CUCCI

Lorenzo CUGLIARI

Elisabetta D'ANASTASIO

Alessia DI CAPRIO

Roberto DI MARTINO

Domenico DI MAURO

Domenico DORONZO

Filippo GRECO

Alessandro IAROCCI

Mario MATTIA

Daniele MELINI

Anna NARDI

Lucia NARDONE

Marco OLIVIERI

Arianna PESCI

Pierangelo ROMANO

Paola SBARRA

Davide SCAFIDI

Maurizio SOLDANI

Sara STOPPONI

Umberto TAMMARO

David ZULIANI

REGISTRAZIONE AL TRIBUNALE DI ROMA N.173 | 2014, 23 LUGLIO

© INGV Istituto Nazionale di Geofisica e Vulcanologia

Rappresentante legale: Presidente INGV

Sede: Via di Vigna Murata, 605 | Roma

Tutti i diritti riservati

www.ingv.it



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA

RAPPORTI TECNICI INGV

PyGraph Scanner
uno strumento interattivo per digitalizzare
serie storiche

PyGraph Scanner
an interactive tool for digitizing historical series

Daniele Randazzo*, Arianna Pesci

INGV | Istituto Nazionale di Geofisica e Vulcanologia, Sezione di Bologna

*Corresponding author

Accettato 24 novembre 2025 | Accepted 24 November 2025

Come citare | How to cite Randazzo, D., & Pesci, A. (2026). PyGraph Scanner uno strumento interattivo per digitalizzare serie storiche. *Rapporti Tecnici INGV*, 513, 1-26. <https://doi.org/10.13127/rpt/513>

Cover Adobe Stock standard license, file n. 1007773045 | In copertina Adobe Stock licenza standard, file n. 1007773045

513

INDICE

Riassunto	7
<i>Abstract</i>	7
Introduzione	8
1. PyGraph Scanner: concetti di base	8
2. Il modello HSV	9
2.1 I vantaggi del formato HSV	10
3. Introduzione all’elaborazione di un grafico	11
3.1 Inserimento dei parametri degli assi	12
3.2 Inserimento punti dello sfondo e del grafico	13
3.3 Tracciamento del grafico	14
4. Salvataggio dei metadata di configurazione	16
5. Salvataggio dei dati	16
6. Esempi e considerazioni	17
7. Dettagli del codice per il tracciamento	18
8. Conclusioni e Ringraziamenti	20
9. Miglioramenti	21
Bibliografia	22
Sitografia	22

Riassunto

Il presente lavoro introduce *PyGraph Scanner*, un software in Python per la digitalizzazione automatica di grafici storici e analogici. Lo strumento nasce dall'esigenza di recuperare serie temporali archiviate su supporto cartaceo, difficilmente utilizzabili con strumenti digitali moderni, e fornisce un ponte tra le osservazioni del passato e le analisi quantitative contemporanee. Il codice utilizza una GUI basata su PySide6, l'elaborazione di immagini tramite OpenCV e la gestione dei dati con NumPy. A partire da pochi punti di riferimento selezionati dall'utente, l'algoritmo segue la traccia del grafico scorrendo in orizzontale e calcolando il centroide dei pixel compatibili per colore, garantendo robustezza al rumore e possibilità di intervento manuale in passaggi complessi.

Il metodo è stato testato su grafici sintetici (rette, sinusoidi e composizioni a frequenze diverse), stampati e fotocopiati per simulare la degradazione dei documenti storici, introducendo volutamente disturbi di fondo. Le prove hanno evidenziato un'elevata affidabilità: le differenze tra curve originali e digitalizzate risultano inferiori allo spessore della traccia, con distribuzioni prossime alla gaussiana e senza alterazioni spettrali significative nelle analisi FFT, salvo le alte frequenze legate al campionamento.

Questi risultati confermano che *PyGraph Scanner* costituisce una soluzione semplice, flessibile e replicabile per la digitalizzazione di serie storiche, particolarmente utile per applicazioni in ambito strutturale, sismico, idrologico e climatico. Sono in corso ulteriori studi sugli errori sistematici e sulle soglie di accettabilità per analisi nel dominio del tempo e della frequenza, che verranno approfonditi in lavori futuri.

Abstract

This work introduces PyGraph Scanner, a Python-based software for the automatic digitization of historical and analog graphs. The tool addresses the need to recover time series stored on paper, which are otherwise difficult to use with modern digital tools, providing a bridge between past observations and contemporary quantitative analyses. The code integrates a GUI built with PySide6, image processing through OpenCV, and data handling with NumPy. Starting from a few reference points selected by the user, the algorithm tracks the graph by scanning horizontally and computing the centroid of color-compatible pixels, ensuring robustness against noise while allowing manual intervention in complex sections.

The method was tested on synthetic graphs (straight lines, sinusoids, and multi-frequency compositions), which were printed and photocopied to simulate the degradation of historical documents and to introduce controlled background disturbances. The experiments demonstrated high reliability: differences between original and digitized curves were consistently smaller than the line thickness, with error distributions close to Gaussian and no significant spectral alterations in FFT analyses, apart from high-frequency components related to dense sampling.

These results confirm that PyGraph Scanner offers a simple, flexible, and reproducible solution for the digitization of historical time series, particularly valuable in structural, seismic, hydrological, and climate applications. Ongoing work focuses on systematic errors and the definition of acceptability thresholds for analyses in both the time and frequency domains, which will be further detailed in future studies.

Keywords Digitalizzazione; Grafico, Serie storiche | Digitalization; Graph; Historical series

Introduzione

La ricerca geodetica si occupa del monitoraggio di strutture di diversa natura, con l'obiettivo principale di valutarne la stabilità, soprattutto in contesti critici. Le sue origini risalgono a epoche antiche, quando il monitoraggio era soprattutto empirico e qualitativo: si basava su ispezioni visive per individuare crepe o inclinazioni, misurazioni manuali rudimentali con filo a piombo o strumenti di livellazione, e perfino sull'ascolto dei suoni per individuare cedimenti nel legno o nel metallo.

A partire dal XVII secolo, figure come Galileo Galilei introdussero concetti scientifici moderni nel calcolo strutturale, studiando la resistenza delle travi; Robert Hooke approfondì l'elasticità dei materiali, mentre Isaac Newton formulò le leggi del moto, ponendo le basi della meccanica dei materiali e delle strutture. Con l'avanzamento della tecnologia, le misurazioni geodetiche divennero sempre più precise. Nel XX secolo nacquero sensori sofisticati come estensimetri, inclinometri, accelerometri, piezometri, che permisero agli ingegneri di ottenere dati quantitativi sul comportamento strutturale e di prevedere potenziali cedimenti.

Tuttavia, molti di questi dati, raccolti per anni o addirittura secoli, sono conservati su supporto cartaceo come grafici e diagrammi, generati da strumentazioni analogiche. La ricerca scientifica moderna si trova quindi a dover valorizzare questa enorme mole di informazioni storiche, indispensabili per comprendere fenomeni complessi e di lungo termine, come i movimenti lenti di strutture monumentali o le deformazioni crostali.

In questo contesto, lo sviluppo di un *software* Python dedicato alla digitalizzazione di serie storiche cartacee non rappresenta solo una semplice conversione, ma costituisce un ponte tra passato e futuro della ricerca. Il programma *PyGraph Scanner*, descritto in questo lavoro, utilizza tecniche avanzate di elaborazione grafica per trasformare immagini di grafici storici in dataset numerici strutturati. In questo modo, dati "dormienti" diventano informazioni attive, pronte per analisi moderne e confronti quantitativi.

Nonostante l'esistenza di numerosi strumenti [1-9] per la digitalizzazione di grafici, da *software* commerciali come UN-SCAN-IT a soluzioni *online* come PlotDigitizer, graph2table, PlotExt, fino a *toolkit open source* come ImageJ o librerie Python (*scikit-image*, *SimpleITK*), nessuno di essi soddisfa pienamente alcuni requisiti fondamentali: fruibilità (interfaccia semplice e utilizzo immediato), modificabilità (codice aperto e adattabile), gratuità, chiarezza (pipeline trasparente e documentata) e soprattutto accuratezza in senso fisico, intesa come capacità di restituire dati digitalizzati quantitativamente attendibili rispetto al disegno originale. Per rispondere a tali esigenze è stato quindi sviluppato un *software open source* in Python, progettato specificamente per la digitalizzazione di serie storiche. Nei capitoli seguenti verrà descritto il funzionamento del *software* e fornita una semplice guida all'uso, permettendo all'utente di operare in autonomia.

Si segnala che questa versione iniziale del pacchetto è principalmente rivolta alla digitalizzazione di dati rappresentati come serie orizzontali, tipiche delle serie temporali. Come esempio illustrativo, verrà utilizzata la serie storica dei movimenti della Torre Garisenda monitorata nel 1904 dal Prof. Francesco Cavani [Cavani, 1903; 1910; 1917] e riscoperti e rianalizzati in tempi recenti [Pesci e Randazzo, 2025] ma senza l'utilizzo di strumenti automatici per l'estrazione dei dati, bensì con paziente tracciamento manuale e notevole impiego di tempo.

1. PyGraph Scanner: concetti di base

L'idea alla base di *PyGraph Scanner* è quella di fornire un algoritmo chiaro, modulare e facilmente leggibile, sviluppato in Python, un linguaggio multiplatforma e ampiamente diffuso nella comunità scientifica. Il *software* combina la semplicità di utilizzo con una struttura scalabile, in

grado di adattarsi a grafici di diverse complessità.

Sul portale GitLab [10] il codice è reso pubblico per mostrare gli algoritmi di elaborazione ed eventualmente permetterne la modifica. Il codice sorgente può essere eseguito da riga di comando in ambiente Python, che è nativo in Linux oppure da installare su Windows e Mac, oppure grazie alla libreria PyInstaller è possibile creare un unico *file* eseguibile, che raccoglie al suo interno tutte le librerie necessarie, ma deve essere creato direttamente nel Sistema Operativo di destinazione.

Per realizzare il codice sono state usate diverse librerie consolidate: PySide6 per creare un'interfaccia grafica (GUI) intuitiva e interattiva, OpenCV per l'elaborazione delle immagini e la gestione dei *pixel*, e NumPy per le operazioni numeriche e la gestione efficiente dei dati.

Partendo da un'immagine scansionata di un grafico, il metodo consente di seguire la traccia in modo semi-automatico, sfruttando pochi punti di riferimento selezionati dall'utente, che identificano sia le coordinate degli assi sia il colore della traccia da seguire. Lo *script* procede *pixel* per *pixel* lungo la direzione orizzontale (tipicamente corrispondente all'asse temporale), analizzando in ciascun passaggio una finestra verticale di *pixel*. All'interno di questa finestra, il codice individua i *pixel* che hanno una tonalità simile a quella dei punti di riferimento selezionati, grazie alla conversione dell'immagine nel modello colore HSV, che separa luminosità, saturazione e tonalità, rendendo più robusta la ricerca anche in presenza di variazioni di illuminazione o rumore di fondo. Quando vengono rilevati più *pixel* appartenenti alla traccia, il software calcola il centroide della regione, ottenendo così una posizione più precisa della traccia del grafico lungo l'asse verticale.

Questo approccio offre diversi vantaggi: 1) permette di estrarre in maniera stabile e ripetibile i valori della traccia anche se la linea è spessa o presenta piccole irregolarità; 2) riduce l'impatto di rumore o artefatti presenti nell'immagine scansionata; 3) mantiene un alto grado di flessibilità, consentendo all'utente di intervenire in punti critici per correggere manualmente eventuali errori o discontinuità.

PyGraph Scanner è concepito per trasformare un'immagine analogica in una serie numerica strutturata, pronta per analisi quantitative, comparazioni con altri dataset e integrazione in studi moderni, per valorizzare e rinnovare dati storici altrimenti difficili da utilizzare.

2. Il modello HSV

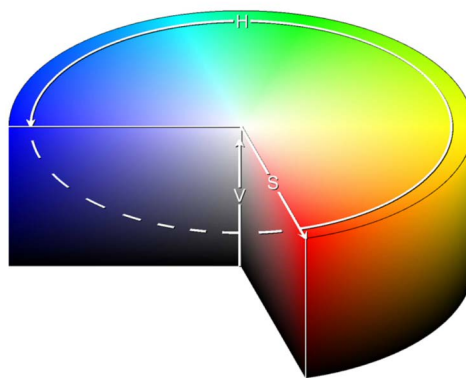
Per rappresentare i colori delle immagini si è scelto il modello HSV, che suddivide ciascun *pixel* in tre componenti:

1. H – *Hue* (tonalità): indica il colore puro, ad esempio rosso, verde o blu;
2. S – *Saturation* (saturazione): misura quanto il colore è intenso o vivido;
3. V – *Value* (luminosità): indica quanto il colore è chiaro o scuro.

A differenza del modello RGB (Rosso, Verde, Blu), che combina luci di base per generare i colori sullo schermo, HSV si avvicina di più alla percezione umana dei colori. Possiamo immaginarlo come un cilindro tridimensionale: lungo il perimetro ruota la tonalità, la saturazione cresce dal centro verso l'esterno, e la luminosità varia lungo l'altezza del cilindro. Questo modello è utile perché permette di distinguere facilmente il colore "puro" dalla sua intensità e luminosità, rendendo immediato interpretare variazioni cromatiche in immagini scientifiche, mappe o rappresentazioni grafiche. In altre parole, HSV trasforma il colore da un concetto tecnico (combinazione di luci) a qualcosa di più intuitivo e vicino a come lo vedono i nostri occhi.

Con riferimento alla Figura 1, il modello HSV rappresenta i colori separando tonalità, saturazione e luminosità, offrendo una lettura più vicina alla percezione umana.

Figura 1 Modello HSV.
Figure 1 HSV model.



La tonalità (*Hue*, *H*) indica il tipo di colore, come rosso, giallo, verde o blu, e può essere immaginata come un angolo che ruota attorno all'asse centrale del cilindro: 0° corrisponde al rosso, 60° al giallo, 120° al verde, 180° al ciano, 240° al blu, 300° al magenta e 360° riporta nuovamente al rosso.

La saturazione (*Saturation*, *S*) misura l'intensità o purezza del colore: vicino al centro del cilindro (0%) i colori appaiono sbiaditi, quasi grigi, mentre verso il bordo esterno (100%) i colori risultano pieni e brillanti.

La luminosità (*Value*, *V*) rappresenta quanto il colore è chiaro o scuro, variando lungo l'asse verticale del cilindro: in basso (0%) il colore tende al nero, in alto (100%) appare luminoso e brillante.

Alcuni esempi pratici di colori in HSV per familiarizzare con questo formato sono in lista:

1. Bianco: $S=0$, $V=255 \rightarrow$ luminoso ma senza tonalità specifica;
2. Nero: $V=0 \rightarrow$ completamente scuro, senza tonalità o saturazione;
3. Grigi: $S=0$, V variabile $\rightarrow$ tonalità assente, varia solo la luminosità;
4. Rosso puro e brillante: $H \approx 0$, $S=255$, $V=255 \rightarrow$ colore intenso e vivo, simile a un fiore acceso;
5. Rosso scuro (bordeaux): $H \approx 0$, $S=255$, $V \approx 100 \rightarrow$ tonalità piena ma scura, colore profondo;
6. Rosso sbiadito (rosa pallido): $H \approx 0$, $S \approx 50$, $V=255 \rightarrow$ tonalità simile ma meno intensa, chiara e delicata.

Il modello HSV consente di separare il colore dalla sua intensità e luminosità, rendendo immediato distinguere variazioni cromatiche e comprendere le caratteristiche visive di ciascun *pixel*. Un approccio particolarmente utile per analisi scientifiche e visualizzazioni di immagini, poiché trasforma dati cromatici complessi in una rappresentazione intuitiva e facilmente interpretabile.

2.1 I vantaggi del formato HSV

Il modello HSV è particolarmente utile per analizzare immagini in cui i colori devono essere riconosciuti con precisione, anche quando la luce cambia o l'immagine non è perfettamente uniforme. Questo perché, a differenza del modello RGB, in HSV la luminosità è separata dalla tonalità e dalla saturazione: mentre in RGB un cambiamento di luce modifica tutti e tre i canali di colore contemporaneamente, in HSV possiamo agire solo sulla tonalità e sulla saturazione senza essere influenzati dal chiarore o dall'ombra. In pratica, questo significa che un rosso brillante alla luce diretta e lo stesso rosso in penombra possono essere trattati come lo stesso colore.

Il modello è anche più intuitivo, perché tonalità, saturazione e luminosità corrispondono in

modo diretto a ciò che percepiamo visivamente. È facile, ad esempio, selezionare una tonalità simile a quella di un grafico, senza preoccuparsi che la luminosità vari a seconda del punto dell'immagine. Inoltre, tonalità e saturazione sono meno sensibili alle variazioni di luce rispetto ai canali RGB, rendendo la rilevazione del colore più stabile e affidabile.

Nel nostro caso pratico, per tracciare i dati di un grafico, il programma permette di scegliere alcuni punti di riferimento sia sul grafico stesso sia sullo sfondo. Calcolando i valori medi di tonalità, saturazione e luminosità di questi punti, il sistema li utilizza come “parametri guida” per trovare automaticamente altri punti con la stessa tonalità. In questo modo, il tracciamento dei dati diventa più preciso e meno influenzato da ombre o cambi di illuminazione.

Per chi vuole approfondire il modello HSV o sperimentare direttamente come i colori si traducono tra HSV e RGB, esistono numerosi tutorial e strumenti online. Ad esempio, è possibile utilizzare un convertitore interattivo che permette di passare facilmente dai tre parametri di tonalità, saturazione e luminosità ai canali rosso, verde e blu, offrendo così una comprensione pratica e immediata [11]. Nel codice descritto in questo lavoro, per facilitare l'individuazione dei colori corretti durante il tracciamento del grafico, si parte dalla selezione di alcuni punti di riferimento sia sul grafico stesso sia sullo sfondo. Il programma calcola quindi i valori medi di tonalità, saturazione e luminosità (HSV) di questi punti del grafico e li utilizza come guida per identificare automaticamente altri punti con la stessa tonalità, permettendo così di trasformare in modo affidabile i dati analogici del grafico antico in serie temporali digitali.

3. Introduzione all'elaborazione di un grafico

Questo paragrafo illustra come interagire con l'interfaccia grafica (GUI) per l'elaborazione dei grafici digitalizzati. All'avvio, il codice si presenta come mostrato in Figura 2.

Il menù “File” consente di eseguire le principali operazioni:

- Apri Immagine: supporta *file* nei formati *.png, *.jpg, *.jpeg, *.bmp, *.tif, *.tiff.
- Salva Configurazione (.cfg): salva i parametri impostati per l'elaborazione, permettendo di ripetere le analisi in modo coerente.
- Salva Risultati Traccia (.csv): esporta le serie temporali ottenute dal tracciamento del grafico.

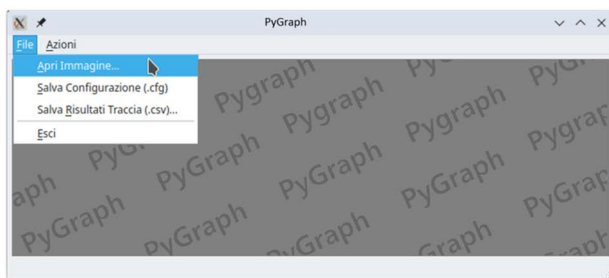


Figura 2 Interfaccia grafica dell'applicazione all'avvio.

Figure 2 Graphical user interface of the application at startup.

Dopo aver selezionato “Apri Immagine” e indicato il *file* da elaborare, il codice verifica se esiste un corrispettivo *file* di configurazione (.cfg) con lo stesso nome dell'immagine. Questo *file* contiene i parametri necessari all'elaborazione (metadata) e può essere creato su richiesta dell'utente tramite la funzione “Salva Configurazione”. Se presente, il codice applica automaticamente i parametri memorizzati, rendendo l'elaborazione più veloce e ripetibile.

Una volta aperta l'immagine, l'utente può navigare agevolmente all'interno del grafico grazie

alla funzione di Zoom (rotellina del mouse) e alla funzione di PAN (Ctrl + click sinistro) per spostare la visualizzazione.

Per testare i limiti dell'algoritmo di tracciamento, è stata utilizzata un'immagine appositamente rumorosa (Figura 3), seguendo la procedura di inserimento dei parametri per verificare la robustezza e l'affidabilità del codice nella ricostruzione delle curve.

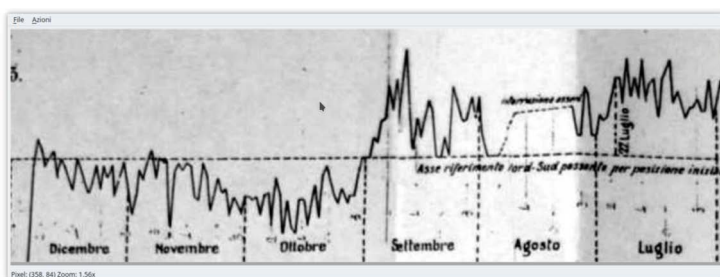


Figura 3 Grafico di prova, con sfondo e grafico rumoroso.
Figure 3 Test graph with noisy background and plot.

3.1 Inserimento dei parametri degli assi

Il primo passo per il tracciamento del grafico consiste nel definire gli assi X e Y, identificando i punti di riferimento (X1, X2) e (Y1, Y2) ai quali sono associati i rispettivi valori della grandezza fisica.

Sull'immagine del grafico viene sovrapposta una finestra di dialogo (Figura 4) in cui inizialmente tutti i parametri sono impostati a zero. L'inserimento delle coordinate avviene in modo interattivo: l'utente posiziona il cursore su uno dei parametri nella finestra e clicca sul grafico per registrare la posizione del punto. La procedura va ripetuta per tutti i punti di riferimento degli assi e, se necessario, le coordinate possono essere anche modificate manualmente.

Una volta definiti i punti ed i rispettivi valori fisici, il codice traccia automaticamente gli assi con una linea rossa. Da queste informazioni calcola il punto origine (intersezione degli assi) e la scala di conversione dai *pixel* della scansione ai valori reali della grandezza fisica.

Nella finestra di dialogo sono visibili sia le coordinate in *pixel* sia i corrispondenti valori della grandezza fisica: ad esempio sull'asse X il punto X2 ha coordinate (411, 150) in *pixel* e corrisponde a 90 giorni, mentre sull'asse Y il punto Y2 ha coordinate (20, 270) e corrisponde a 4 mm di spostamento. In questo modo, il codice può trasformare ogni punto dell'immagine in un dato numerico utilizzabile per le serie temporali.

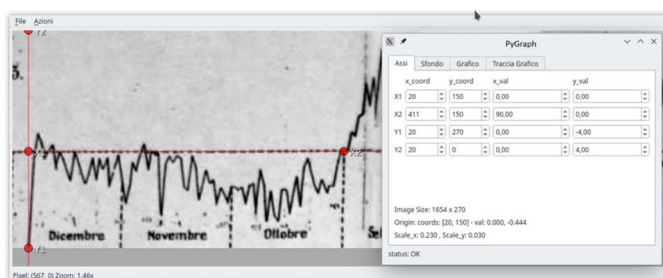


Figura 4 Definizione degli assi XY, coordinate e valori associati.
Figure 4 Definition of the XY axes, coordinates, and associated values.

3.2 Inserimento punti dello sfondo e del grafico

Al secondo *tab* della finestra di dialogo viene chiesto di inserire i punti dello sfondo. Anche qui l'inserimento è interattivo, cliccando sul grafico i punti vengono numerati e aggiunti alla lista, con possibilità di essere selezionati e cancellati all'occorrenza usando i rispettivi bottoni, così come mostrato in Figura 5. Notiamo che l'insieme dei punti selezionati con le loro caratteristiche HSV vengono riportati in tabella con la relativa statistica in basso che, se accettata dall'operatore, diventerà l'elemento di confronto per la ricerca della traccia nelle fasi successive. Inoltre, una valutazione preliminare dei valori di HSV serve all'operatore per vedere con maggiore dettaglio il disturbo di uno sfondo che per una tavola antica potrebbe rappresentare un problema e necessitare di un pretrattamento di immagine.

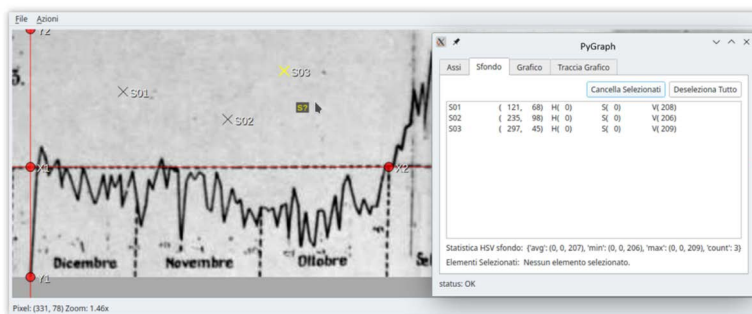


Figura 5 Selezione dei punti dello sfondo.

Figure 5 Selection of background points.

Analogamente a quanto visto per lo sfondo, è necessario inserire alcuni punti di riferimento del grafico. Il numero dei punti scelto ha diverse implicazioni: delimita la zona del grafico da digitalizzare, definisce i punti di passaggio "obbligatori" utili nel caso di interruzioni del grafico, e attiva l'elaborazione con un numero di *processi paralleli* pari al numero dei segmenti individuati ovvero $n-1$ punti del grafico, ciò rende l'elaborazione molto più efficiente soprattutto per i grafici scansionati con risoluzione elevata. In Figura 6 è riportata la selezione di due punti per mostrare il comportamento del codice di tracciamento.

In questa finestra di dialogo dedicata ai punti del grafico è presente anche il bottone "Connetti Selezionati", per collegare un punto al successivo con un segmento diretto, è un bottone di tipo *toggle* (ovvero *on/off*) che attiva la funzione *connesso* ai punti selezionati e facilita il tracciamento (sarà approfondito in seguito).

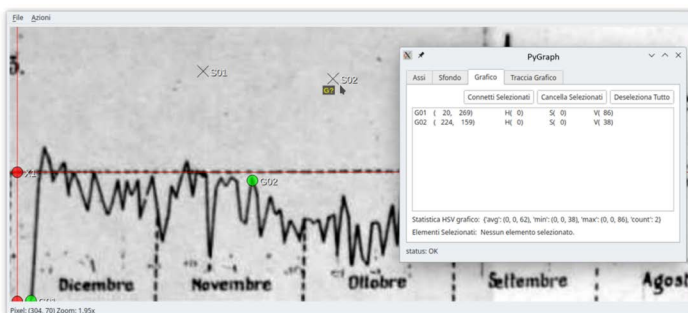


Figura 6 Inserimento dei punti del grafico.

Figure 6 Input of the graph points.

3.3 Tracciamento del grafico

Il quarto *tab* della finestra di dialogo serve a definire i parametri per il tracciamento del grafico, in particolare i valori minimo e massimo dei parametri HSV, il passo dei *pixel* “Step X Trace” per la ricerca lungo l’asse X e l’ampiezza “Delta Y Trace” della finestra di *pixel* in cui cercare la traccia.

Sulla destra sono mostrati i rispettivi valori medi dei parametri HSV del grafico e dello sfondo, mentre il bottone “Imposta HSV” propone i parametri HSV a partire dai punti selezionati del grafico, aggiungendo un margine di tolleranza.

Il bottone “Elabora Traccia Grafico” avvia l’elaborazione dell’immagine così come mostrato in Figura 7 per una piccola porzione del grafico usando i parametri di *default* (modificabili dall’operatore).

Prima di procedere alla descrizione dei parametri della figura, bisogna tenere presente la statistica riportata in Figura 6 che mostra i seguenti valori per HSV:

- Media: (0, 0,62)
- Minimo: (0, 0, 38)
- Massimo: (0, 0, 86)

I parametri di minimo e massimo che contengono ogni sfumatura rilevata, saranno utilizzati come base di partenza per ampliare l’intervallo di ricerca in un margine iniziale di ± 10 unità, come riportato nella finestra di dialogo in Figura 7.

I valori di H, che in Figura 1 giacciono sul perimetro del cilindro e che nella rappresentazione Python hanno valori fra 0 e 180 (angolo piatto), diventano $H_{\min} = 180 - 10 = 170$ e $H_{\max} = 0 + 10 = 10$;

L’intervallo originale di S diventa $S_{\min} = 0$ e $S_{\max} = 10$, mentre l’intervallo V diventa $V_{\min} = 28$ e $V_{\max} = 96$. Si ricorda che tutti i parametri scelti e impostati possono essere sempre modificati e salvati per le elaborazioni successive e per scopo di confronto tra i risultati.

Un altro parametro essenziale è Delta_Y Trace (in figura pari a 30), è l’intervallo in *pixel* della colonna verticale della matrice immagine; se impostato con un valore troppo piccolo non permette di seguire pendenze esasperate del grafico, comportando la perdita della traccia. In questo caso il codice attribuisce l’ultimo valore trovato, restituendo una linea fittizia orizzontale di collegamento col primo punto utile successivo. Questa opzione è stata implementata per limitare l’area di ricerca intorno al grafico, ottimizzare i tempi di esecuzione e prevenire l’intercettazione di rumori di fondo che creerebbero una traccia non reale.

Infine, seppur scontato per completezza si informa che Delta_X Trace è l’intervallo di campionamento lungo l’asse orizzontale, che per una migliore risoluzione generalmente si imposta al valore 1, altrimenti può essere allargato in presenza di curve morbide e lente.

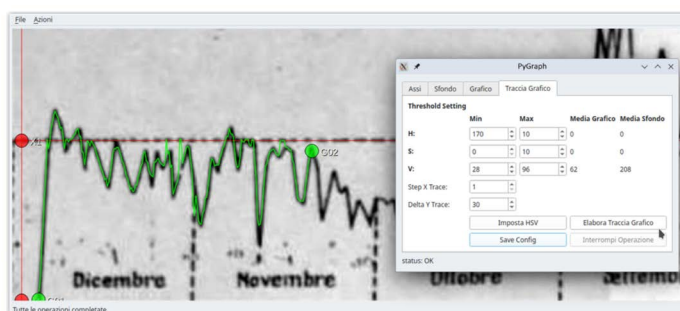


Figura 7 Finestra di dialogo e parametri per il tracciamento del grafico.
Figure 7 Dialog window and parameters for graph tracing.

Come si può osservare la traccia presenta delle imperfezioni laddove l'immagine è più rumorosa e, estendendo ulteriormente la ricerca dei punti, la Figura 8 mostra altre zone in cui il tracciamento del grafico si perde fra le indicazioni disegnate sulla immagine come linee tratteggiate, scritture e rumore in generale, ciò accade perché quei *pixel* selezionati dal tracciamento automatico, anche se errati, rientrano nel criterio di selezione del centroide.

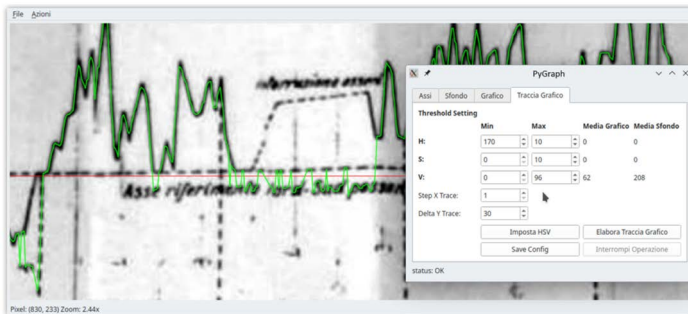


Figura 8 Tracciamento errato del grafico a causa del rumore nell'immagine.

Figure 8 Errors in graph tracing due to image noise.

Per correggere gli errori dovuti alla selezione automatica, il codice offre la possibilità di scegliere dei punti di passaggio obbligati, che sono di due tipi: il punto di tipo “semplice” ovvero di passaggio che può essere usato anche per creare un nuovo segmento da elaborare con un processo dedicato, oppure un punto “connesso” al successivo ovvero da congiungere direttamente nel caso in cui i punti dell'immagine non permettano il tracciamento. La Figura 9 mostra entrambi i casi di punti semplici e connessi mentre la finestra di dialogo mostra l'elenco dei punti. Si nota nella figura che l'asse orizzontale dell'immagine originale è deformato, questo perché al momento dello scatto della foto la tavola cartacea non era perfettamente in piano. Ciò non viene preso in considerazione in questo lavoro in quanto lo scopo è mostrare il funzionamento dell'algoritmo nella ricerca della traccia.

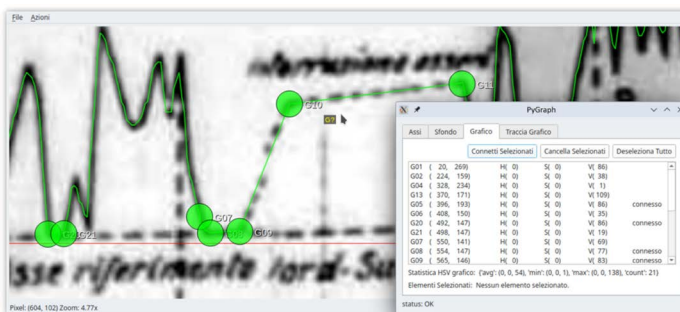


Figura 9 Punti semplici e punti connessi.

Figure 9 Simple points and connected points.

La Figura 10 mostra il tracciamento esteso a tutta la sequenza, in cui è stato necessario intervenire aggiungendo solo 28 punti del grafico per recuperare le zone più rumorose su 1499 punti tracciati, come riportato nella finestra di dialogo. Il tempo di esecuzione è dell'ordine del secondo e, considerando l'intervento di preparazione, configurazione e scelta dei parametri di ricerca la digitalizzazione ha comportato un totale di alcuni minuti di lavoro.

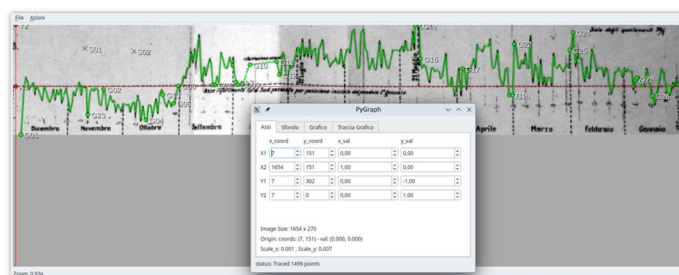


Figura 10 Tracciamento di tutta la sequenza.
Figure 10 Tracking of the entire series.

4. Salvataggio dei metadata di configurazione

Il salvataggio dei parametri di configurazione avviene dal menu “File → Salva Configurazione (.cfg)” oppure mediante il bottone nella finestra di dialogo “Traccia Grafico → Save Config”. Questo *file* contiene tutti i parametri inseriti nella finestra di dialogo ed è un *file* leggibile in formato di tipo JSON, che permette anche la modifica manuale direttamente dal *file* (rispettando la sintassi). La Figura 11 riporta un estratto del *file* in cui si leggono le coordinate dei punti di riferimento X1 e X2.

```
{
  "Name": "/media/dr/Data/Projects/008 PyGraph_Scanner/PyGraph_Scanner/DataIn/grafico prova CAVANI.jpg",
  "axis": {
    "X1": {
      "coords": {
        "_qpointf_": true,
        "x": 7.0,
        "y": 151.0
      },
      "spin_val_x": 0.0,
      "spin_val_y": 0.0
    },
    "X2": {
      "coords": {
        "_qpointf_": true,
        "x": 874.0,
        "y": 151.0
      },
      "spin_val_x": 1.0,
      "spin_val_y": 0.0
    }
  }
}
```

Figura 11 Estratto dal *file* di configurazione.
Figure 11 A part of the configuration file.

5. Salvataggio dei dati

Una volta tracciato il grafico, è possibile esportarlo attivando il menu “File → Salva Risultati Traccia (.csv)”, questa funzione salva la lista delle coordinate in *pixel* (colonne 1 e 2) ed i rispettivi valori (colonne 3 e 4) in un *file* “.csv” così come mostrato in Figura 12, dove viene riportato un estratto del *file* esportato. Per convertire i *pixel* del grafico al rispettivo valore, il codice esegue la “Calibrazione Assi” ovvero conoscendo il punto di origine dall’incrocio degli assi, e conoscendo i valori reali (metrici) associati ai punti X1, X2, Y1, Y2, si ricava il fattore di scala che permette di calcolare il valore del punto sul grafico mediante proporzione diretta. La Figura 12 mostra la porzione di un *file* di dati esportato dopo l’elaborazione, in cui l’origine degli assi è alla coordinata (7, 151). L’asse delle ascisse X è individuato da tutti i punti con ordinata Y=151, questo ci permette di fare una prima validazione del tracciamento; infatti, così come

evidenziato alle righe 21, 23 e 24, il valore del grafico all'incrocio con l'asse X assume proprio il valore $y_val = "0.0"$.

1	x_coord,y_coord,x_val,y_val
2	20,263,0.007893139040680024,-0.7417218543046358
3	21,253,0.008500303582270797,-0.6754966887417219
4	22,234,0.009107468123861567,-0.5496688741721855
5	23,216,0.009714632665452338,-0.4304635761589404
6	24,196,0.010321797207043109,-0.2980132450331126
7	25,178,0.01092896174863388,-0.17880794701986755
8	26,168,0.011536126290224652,-0.11258278145695365
9	27,150,0.012143290831815423,0.006622516556291391
10	28,150,0.012750455373406194,0.006622516556291391
11	29,142,0.013357619914996965,0.059602649006622516
12	30,137,0.013964784456587736,0.09271523178807947
13	31,131,0.014571948998178506,0.13245033112582782
14	32,131,0.015179113539769279,0.13245033112582782
15	33,132,0.015786278081360048,0.12582781456953643
16	34,134,0.01639344262295082,0.11258278145695365
17	35,138,0.017000607164541593,0.08609271523178808
18	36,141,0.017607771706132362,0.06622516556291391
19	37,142,0.018214936247723135,0.059602649006622516
20	38,150,0.018822100789313904,0.006622516556291391
21	39,151,0.019429265330904676,0.0
22	40,156,0.02003642987249545,-0.033112582781456956
23	41,151,0.020643594414086218,0.0
24	42,151,0.02125075895567699,0.0
25	43,152,0.02185792349726776,-0.006622516556291391
26	44,150,0.022465088038858532,0.006622516556291391

Figura 12 Estratto dei dati dal file esportato ".csv".

Figure 12 A part of exported ".csv" data.

6. Esempi e considerazioni

Per valutare il comportamento del codice in condizioni non critiche, ossia in assenza di immagini eccessivamente rumorose, sono state elaborate immagini standard di riferimento (rette inclinate, sinusoidi a diversa frequenza e forme d'onda composte). È stato sufficiente selezionare i soli punti iniziale e finale del grafico: l'elaborazione è proseguita automaticamente senza la necessità di punti intermedi (Figura 13).

La figura, in alto, mostra alcuni esempi: rette, sinusoidi ad alta e bassa frequenza e una composizione più complessa, poi analizzata. La traccia digitalizzata dal software (in verde) risulta perfettamente sovrapposta alla curva originale (in nero). La serie più complessa, composta dalle precedenti, è stata analizzata e, a destra della figura, è riportata la statistica (istogramma delle differenze tra originale e digitalizzato e interpolazione con gaussiana) e la relativa analisi FFT (utilizzata per verificare l'eventuale introduzione di frequenze spurie). Dalle analisi si rileva che:

1. i residui, ossia le differenze tra curva nota e curva digitalizzata, mostrano solo sistematismi trascurabili, con deviazioni standard inferiori allo spessore della linea;
2. la distribuzione delle differenze è assimilabile a una gaussiana;
3. i grafici nel dominio delle frequenze non evidenziano anomalie, se non i tipici effetti di *aliasing* o di apparenti alte frequenze. In generale, una risoluzione più elevata dell'immagine riduce l'errore nel tracciamento, mentre la presenza di griglie o di *pixel* dello stesso colore in prossimità della curva può introdurre piccole incertezze.

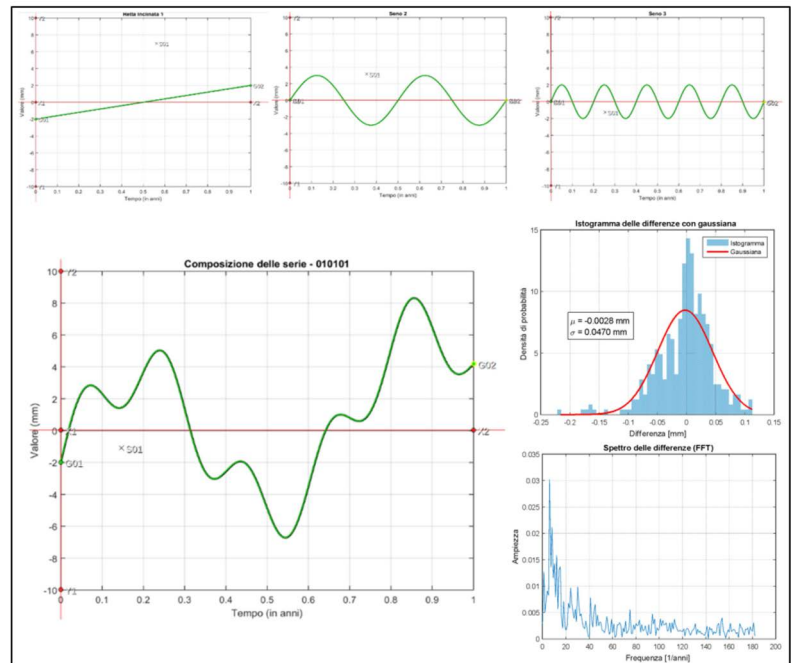


Figura 13 Tracciamento di curve e forme d'onda semplici e complesse; esempio di valutazione dei risultati.

Figure 13 Tracing of simple and complex curves and waveforms; example of result evaluation.

Per testare e validare il software *PyGraph Scanner* sono state inoltre prodotte immagini “analogiche” con serie temporali artificiali (rette con diversi coefficienti angolari, sinusoidi a frequenze variabili, combinazioni di tratti piatti e ripidi). Alcune sono state stampate e fotocopiate, per introdurre artificialmente caratteristiche tipiche di documenti storici: variazioni di contrasto, deformazioni della carta, macchie o scritte di fondo. In tutti i casi, la traccia principale è stata rilevata correttamente, dimostrando la robustezza dell’algoritmo rispetto a rumore e disturbi superficiali. Le analisi di confronto tra le curve originali e quelle digitalizzate hanno mostrato un’elevata concordanza: gli errori riscontrati risultano quasi sempre contenuti nello spessore della linea del grafico, sia nel dominio temporale sia in quello frequenziale. Le griglie possono generare interferenze locali, ma senza compromettere in modo significativo la qualità dei dati. Le analisi FFT hanno evidenziato solo rare componenti spurie ad alta frequenza, riconducibili alla densità di campionamento dei *pixel* più che ai limiti dell’algoritmo.

I test confermano dunque che *PyGraph Scanner* può trattare efficacemente immagini analogiche anche “invecchiate” o parzialmente degradate, restituendo serie digitali fedeli e pronte per analisi quantitative.

Tutti i possibili errori sistematici, distorsioni dovute al campionamento, alterazioni introdotte da fotocopie o macchie di fondo, sono attualmente oggetto di studio. In un prossimo lavoro verrà presentato un approccio mirato a definire soglie di accettabilità per l’analisi delle serie temporali digitalizzate, con particolare attenzione agli effetti sulle componenti frequenziali, insieme allo sviluppo di metodi di correzione automatica per garantire un’interpretazione affidabile dei dati storici trasformati in digitale.

7. Dettagli del codice per il tracciamento

Questo codice (pubblicato su GitLab [10]) segue una logica ad oggetti per gestire la GUI e le interazioni con l’utente, ma la parte che definisce il tracciamento dei punti è essenzialmente composta da un ciclo che, ad ogni punto (x, y) associato al grafico, attiva la selezione dei *pixel*

all'interno di una finestra verticale, e su tali punti calcola il centroide.

La Figura 14 mostra la funzione che scansiona i *pixel*, essa è denominata “*find_centroid_match_vertically*” e individua i *pixel* che rientrano nei limiti del filtro HSV all'interno di una finestra verticale “Delta_Y”. Alla funzione vengono passati i parametri *hsv_image* (immagine in formato HSV), le coordinate (cx, cy) dell'ultimo punto trovato, l'ampiezza della finestra verticale “dy” ed i valori min e max dei parametri HSV.

```

2539 def find_centroid_match_vertically(hsv_img, cx, cy, dy, 1 usage ± randazzo +1 *
2540                                     h_min, h_max, s_min, s_max, v_min, v_max):
2541     h, w, _ = hsv_img.shape
2542     ymin = max(0, cy - dy)
2543     ymax = min(h - 1, cy + dy)
2544     search_x = max(0, min(w - 1, cx))
2545     matches = []
2546     for y in range(ymin, ymax + 1):
2547         hsv_px = hsv_img[y, search_x]
2548         if check_pixel_match(
2549             hsv_px,
2550             h_min, h_max,
2551             s_min, s_max,
2552             v_min, v_max):
2553             matches.append((search_x, y))
2554     return calculate_centroid(matches)

```

Figura 14 Funzione per filtrare i punti in base ai parametri HSV.

Figure 14 Function for filtering points based on HSV parameters.

Ad ogni ciclo le coordinate del *pixel* in esame ed i suoi parametri HSV vengono passati dalla funzione “*check_pixel_match*” (riga 2548) per verificare la corrispondenza con i criteri di selezione, così come mostrato in Figura 15.

```

2664 def check_pixel_match(hsv_pixel, 1 usage ± randazzo +1 *
2665                       h_min, h_max, s_min, s_max, v_min, v_max):
2666
2667     h_px, s_px, v_px = map(int, hsv_pixel)
2668
2669     if h_min <= h_max:      # Caso 1: Intervallo normale (es. h_min=10, h_max=30)
2670         h_in_range = h_min <= h_px <= h_max
2671     else:                  # Caso 2: Intervallo circolare (attraversa lo zero es. h_min=170, h_max=10)
2672         h_in_range = (h_px >= h_min) or (h_px <= h_max)
2673
2674     if not h_in_range: return False
2675     if not (s_min <= s_px <= s_max): return False
2676     if not (v_min <= v_px <= v_max): return False
2677     return True

```

Figura 15 Confronto dei parametri HSV per il singolo *pixel*.

Figure 15 Comparison of HSV parameters for a single pixel.

Se il *pixel* elaborato rientra nei criteri di selezione, le coordinate del punto vengono aggiunte alla lista denominata “*matches*” (riga 2553 del codice in Figura 14). quando la scansione della finestra verticale è completa, la lista dei punti selezionati viene passata alla funzione che ne seleziona il centroide.

Il calcolo del centroide viene effettuato dalla funzione mostrata in Figura 16, denominata “*calculate_centroid*”: essa riceve come parametro di ingresso la lista dei punti contenuta nell’intervallo Delta_Y, con i rispettivi parametri HSV, in cui ci si aspetta di trovare la traccia del grafico. Da questa lista di parametri viene ricavato il minimo valore del parametro “V” associato al punto più scuro (riga 2595) e successivamente viene estratta la lista dei punti “y” con quel valore minimo (riga 2604). In fine dalla lista di queste ordinate “y” viene selezionato il punto mediano, che viene poi restituito sotto forma di coordinate (x, y). I commenti aggiunti al codice aiutano a seguire il flusso della procedura.

La scelta della libreria NumPy permette di gestire in modo più efficiente le liste, rendendo anche il codice più leggibile.

```

2585 def calculate_centroid(points): # usage ± randazzo *
2586     if not points: return None
2587
2588     # Converti la lista in un array NumPy per una gestione più semplice
2589     points_array = np.array(points, dtype=object)
2590
2591     # Estrai i valori 'V', che sono il terzo elemento della lista HSV
2592     v_values = np.array([hsv[2] for hsv in points_array[:, 2]])
2593
2594     # Trova il valore V minimo
2595     min_v = np.min(v_values)
2596
2597     # Seleziona gli indici di tutti i punti che hanno questo valore minimo
2598     min_v_indices = np.where(v_values == min_v)[0]
2599
2600     # Estrai i punti corrispondenti a questi indici
2601     darkest_points = points_array[min_v_indices]
2602
2603     # Estrai le coordinate Y da questi punti
2604     y_coords_of_darkest = darkest_points[:, 1]
2605
2606     # Calcola la mediana delle coordinate Y per trovare il valore centrale
2607     median_y = int(np.median(y_coords_of_darkest))
2608
2609     # Seleziona il punto centrale più scuro dalla lista di punti scuri
2610     central_darkest_point = tuple((points[0][0], median_y))
2611
2612     return tuple(central_darkest_point)

```

Figura 16 Funzione per il calcolo del centroide.

Figure 16 Function for calculating the centroid.

8. Conclusioni e Ringraziamenti

Prima dello sviluppo di questo *software*, è stata condotta una ricerca per verificare l’esistenza di strumenti analoghi con interfaccia grafica (GUI). È emerso che PyGraph offre un approccio semplice, flessibile e replicabile per il tracciamento di grafici storici e analogici.

Dall’analisi del suo funzionamento, risulta chiaro che la scelta accurata dei parametri, dagli assi XY ai punti di riferimento (grafico e sfondo), fino ai filtri HSV, assicura un metodo solido e versatile.

I principali punti di forza del codice sono:

1. Robustezza al rumore verticale: il calcolo del centroide permette di rilevare la coordinata

- Y in modo stabile, anche in presenza di linee spesse o piccoli artefatti.
2. Rilevazione basata sul colore: la distinzione tra traccia e sfondo facilita l'estrazione dei dati, anche con variazioni di luminosità o contrasto.
 3. Elaborazione *multi-thread*: il processamento parallelo accelera i tempi di elaborazione e rende il sistema più reattivo.
 4. Intervento localizzato: l'operatore può connettere manualmente punti critici, migliorando il tracciamento nei passaggi complessi.

Attualmente, PyGraph è ottimizzato per tracciamento orizzontale da destra verso sinistra, escludendo linee verticali o retrograde. Tuttavia, futuri sviluppi potrebbero superare limiti legati a linee dello stesso colore sovrapposte, linee spesse o irregolari, sfondi complessi o assi inclinati.

L'applicazione storica scelta per lo sviluppo del codice riguarda il monitoraggio della Torre Garisenda di Bologna da parte del Prof. Cavani (1900 circa). Questo esempio dimostra come i grafici storici rappresentino un patrimonio informativo cruciale: pur analogici, contengono dati fondamentali per ricostruire fenomeni passati, validare modelli e analizzare eventi sismici, climatici o strutturali.

Altri ambiti di applicazione includono: monitoraggio sismico e vulcanico con sismogrammi e inclinometri storici; idrologia e climatologia con registrazioni di livelli di fiumi, piogge e temperature; ingegneria strutturale e conservazione dei beni culturali; oceanografia e geofisica marina con misure storiche di maree, temperatura e salinità.

PyGraph consente di trasformare dati provenienti da registrazioni analogiche in serie digitali, offrendo la possibilità di sfruttare informazioni storiche in chiave moderna. Questa conversione apre la strada a ulteriori analisi quantitative, comparazioni con dati contemporanei e modellazione predittiva, rendendo fruibili informazioni altrimenti difficili da trattare con strumenti digitali.

PyGraph si distingue per semplicità, robustezza e flessibilità: particolarmente efficace per grafici storici o complessi con sufficiente contrasto, e facilmente integrabile con tecniche avanzate in caso di grafici rumorosi o con linee intersecanti.

9. Miglioramenti

Possibili sviluppi futuri potrebbero prevedere la possibilità di inserire il codice in un'unica immagine Docker eseguibile, che contenga già preinstallate tutte le librerie necessarie in modo da avere una applicazione pronta per l'uso. Altri possibili scenari potrebbero essere l'utilizzo della libreria Param, che è un potente strumento Python focalizzato sulla gestione dei parametri e dei loro attributi, soprattutto nel contesto di applicazioni scientifiche, di data science, o interattive, essa permette la Validazione e Vincoli Automatici sui parametri, un Sistema di Documentazione e Metadati e la Gestione degli Eventi (*Watcher*). Un'altra libreria utile potrebbe essere Panel, un altro potente strumento Python che permette di creare dashboard interattive e applicazioni web standalone combinando facilmente oggetti di visualizzazione, widget, e codice Python reattivo, essa si appoggia alla libreria Param. Ancora un'altra libreria utile potrebbe essere Xarray, che estende le capacità di NumPy fornendo strutture dati e strumenti per la gestione efficiente e l'analisi di dati multidimensionali etichettati (labelled multi-dimensional arrays), in particolare nel contesto della scienza dei dati e della geoscienza. Di queste librerie è possibile trovare in rete ampia documentazione al riguardo.

Si ringrazia il revisore per aver dato spunto ai miglioramenti ed aver condiviso la sua esperienza nell'uso di specifiche librerie Python.

Bibliografia

- Pesci, A., & Randazzo, D. (2025). Le serie temporali storiche della Torre Garisenda (1904). *Quaderni di Geofisica INGV*, 198, 1-34. <https://doi.org/10.13127/qdg/198>
- Cavani, F. (1903). *Pendenza, stabilità e movimenti delle Torri Garisenda e Ghirlandina*. Collegio degli Ingegneri e Architetti di Bologna, Bologna.
- Cavani, F. (1910). *Movimenti della sommità rispetto alla base nella torre Garisenda di Bologna. Memoria letta nella seduta del 22 maggio 1910*. Collegio degli Ingegneri e Architetti di Bologna, Bologna. (con tavola).
- Cavani, F. (1917). *Sulla pendenza delle principali torri di Bologna, Modena e Pisa*. Collegio degli Ingegneri e Architetti di Bologna, Bologna.

Sitografia

- [1] <https://www.im2graph.co.il>
- [2] <https://plotdigitizer.com>
- [3] <https://www.silkscientific.com/graph>
- [4] <https://graph2table.com>
- [5] <https://plotext.com>
- [6] <https://getdata-graph-digitizer.com>
- [7] <https://www.digitizeit.xyz>
- [8] <https://imagej.nih.gov/ij/>
- [9] <https://arxiv.org/abs/1502.05241>
- [10] <https://gitlab.rm.ingv.it/daniele.randazzo/pygraph>
- [11] www.rapidtables.com/convert/color/hsv-to-rgb.html



QUADERNI di GEOFISICA

ISSN 1590-2595

<https://collane-editoriali.ingv.it/index.php/qdg/it/index/>

QUADERNI di GEOFISICA è una rivista scientifica che pubblica dati, misurazioni e osservazioni geofisiche (anche preliminari) in italiano e inglese. L'obiettivo è una diffusione rapida e gratuita per la comunità scientifica, garantita da un processo di peer-review e da un Editorial Board multidisciplinare. La rivista è inclusa in "Emerging Sources Citation Index" di Clarivate Analytics e "Open Access Journals" di Scopus, e adotta il modello Diamond Open Access, che non prevede costi né per gli autori né per i lettori.

QUADERNI di GEOFISICA is a scientific journal that publishes geophysical data, measurements and observations, including preliminary results, in Italian and English. Its aim is to ensure rapid and free dissemination to the research community, supported by a peer-review process and by a multidisciplinary Editorial Board. The journal is included in the Emerging Sources Citation Index of Clarivate Analytics and it's included in the Open Access journals of Scopus. It follows the Diamond Open Access model, with no charges for either authors or readers.



RAPPORTI TECNICI INGV

ISSN 2039-7941

<https://collane-editoriali.ingv.it/index.php/rt/it/index>

RAPPORTI TECNICI INGV è una rivista tecnico-scientifica che mette in mostra le numerose attività tecnologiche e di ricerca condotte dai ricercatori e tecnici INGV, e non solo. Pubblicata in italiano e inglese, RAPPORTI TECNICI INGV è ideata per valorizzare anche quei lavori che offrono un contributo significativo dal punto di vista tecnico, operativo o documentale fondamentale per lo sviluppo dell'Ente. La rivista adotta il modello Diamond Open Access, che non prevede costi né per gli autori né per i lettori.

RAPPORTI TECNICI INGV is a technical-scientific journal that showcases the wide range of technological and research activities carried out by INGV researchers and technical staff, as well as by external contributors. Published in Italian and English, RAPPORTI TECNICI INGV is designed to highlight works that provide a significant contribution from a technical, operational, or documentary perspective, and that are essential to INGV development. The journal follows the Diamond Open Access model, with no charges for either authors or readers.



MISCELLANEA INGV

ISSN 2039-6651

<https://collane-editoriali.ingv.it/index.php/mi/it/index>

MISCELLANEA INGV rappresenta una tipologia di editoria non tradizionalmente catalogabile nelle ordinarie riviste scientifiche di settore. Pubblicata in lingua italiana e inglese, MISCELLANEA INGV offre uno spazio per la pubblicazione di materiali eterogenei che, pur non rientrando nei formati più strutturati delle altre Collane editoriali INGV, rivestono un interesse scientifico, tecnico o divulgativo. La rivista adotta il modello Diamond Open Access, che non prevede costi né per gli autori né per i lettori.

MISCELLANEA INGV represents a form of publishing that cannot be classified within the standard scientific journals. Published in Italian and English, MISCELLANEA INGV provides a venue for the publication of heterogeneous materials that are of scientific, technical or outreach interest, even though they do not fall within the more structured formats of the other INGV editorial series. The journal follows the Diamond Open Access model, with no charges for either authors or readers.

Progetto grafico
Barbara ANGIONI
Istituto Nazionale di Geofisica e Vulcanologia

Ufficio Editoriale
Francesca DI STEFANO | Coordinatore

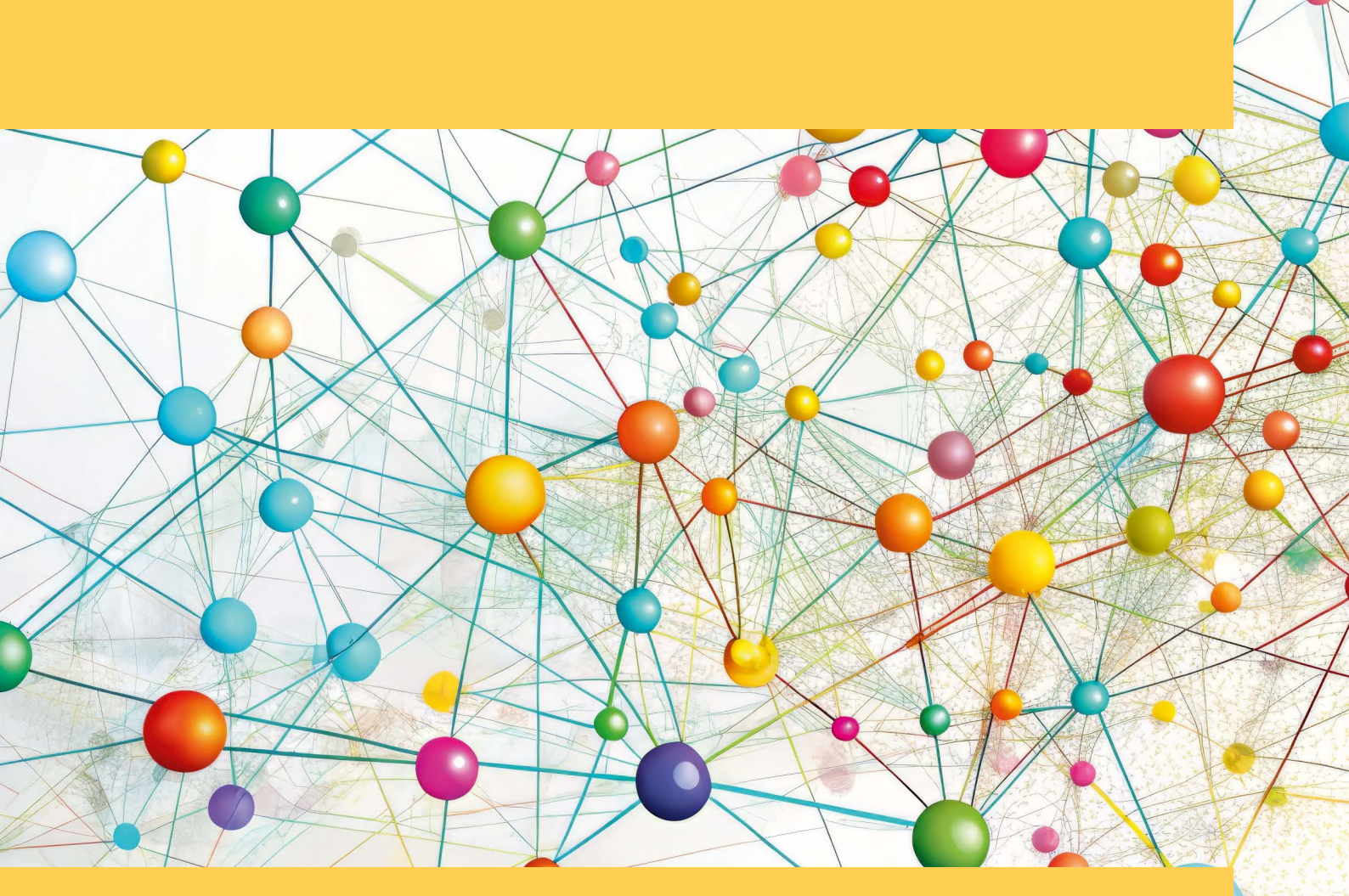
Segreteria di redazione
segreteria.collane-editoriali@ingv.it

Editing | Produzione | Grafica redazionale
Barbara ANGIONI
Massimiliano CASCONI
Francesca DI STEFANO
Patrizia PANTANI

© 2026 the Author(s). All rights reserved. Open Access



Creative Commons Attribution 4.0 International



ISTITUTO NAZIONALE DI GEOFISICA E VULCANOLOGIA