



# UNIVERSITÀ DEGLI STUDI DELLA BASILICATA

DIPARTIMENTO DI INGEGNERIA

CORSO DI LAUREA MAGISTRALE IN INGEGNERIA INFORMATICA E DELLE TECNOLOGIE DELL'INFORMAZIONE

## RELAZIONE PROGETTO

Sensori, Rivelatori e Dispositivi Elettronici

Docente:

Prof. Antonio Iula

Studente:

Michael Pio Stolfi 68787

## ANNO ACCADEMICO 2024-2025

# Sommario

# 1. Introduzione

Il metodo denominato CCM (dai suoi creatori Capece, Caporale, Manfreda), è un metodo di feature extraction utile nell'estrazione del pattern venoso del palmo di una mano. Insieme con il presente documento è presente anche il manuale originale del metodo, che però in alcuni punti, risulta poco chiaro e oltremodo ostico.

Lo scopo principe del presente elaborato è quello di riassumere, analizzare e presentare in maniera composta e ordinata i risultati ottenibili con il "metodo CCM".

Infine viene proposta una analisi critica del metodo CCM, in cui si entra nel merito delle scelte implementative più discutibili. Per ognuna di queste viene spiegato il perché è discutibile, dove potrebbe non funzionare o funzionare male e come potrebbe essere migliorata.

# 2. Il metodo CCM

## 2.1. Dati di partenza

Il metodo CCM prende come input inziale i file "mat" contenenti i dati relativi alle varie acquisizioni dei vari utenti. È possibile osservare il contenuto di un file mat nella cattura seguente:

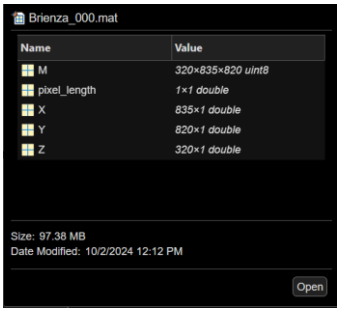



*Figura 1: Preview MATLAB di un file mat relativo all'acquisizione "00" dell'utente "Brienza".*

Delle matrici precedenti quella di interesse è la matrice M, che come è possibile osservare è una matrice tridimensionale di dimensioni originali 835×820×320 voxel. Per tale matrice, come è possibile notare anche dalle dimensioni dei singoli vettori colonna, la convenzione utilizzata per le dimensioni è la seguente:

- 1ª dimensione (i) → righe → contiene i dati dell'asse Z;
- 2ª dimensione (j) → colonne → contiene i dati dell'asse X;
- 3ª dimensione (k) → pagine → contiene i dati dell'asse Y.

Quindi la matrice di partenza è nella forma $zxy$, però molte funzioni di elaborazione e stampa delle immagini tridimensionali in MATLAB usano la forma $yxz$. Il passaggio dalla prima alla seconda forma è banale infatti basta eseguire il seguente pezzo di codice:

$$M = permute(M, [3\ 2\ 1]);$$

La precedente matrice può essere ora comodamente mostrata in MATLAB grazie all'ausilio della funzione predefinita "$volshow(\dots)$". Per mostrare questo volume e tutti quelli seguenti verrà utilizzata la funzione custom "*graficoVolshow(…)*" che di occupa in autonomia della creazione di alcuni componenti grafici necessari a lanciare $volshow$. La matrice tridimensionale di partenza si presenta quindi come segue:

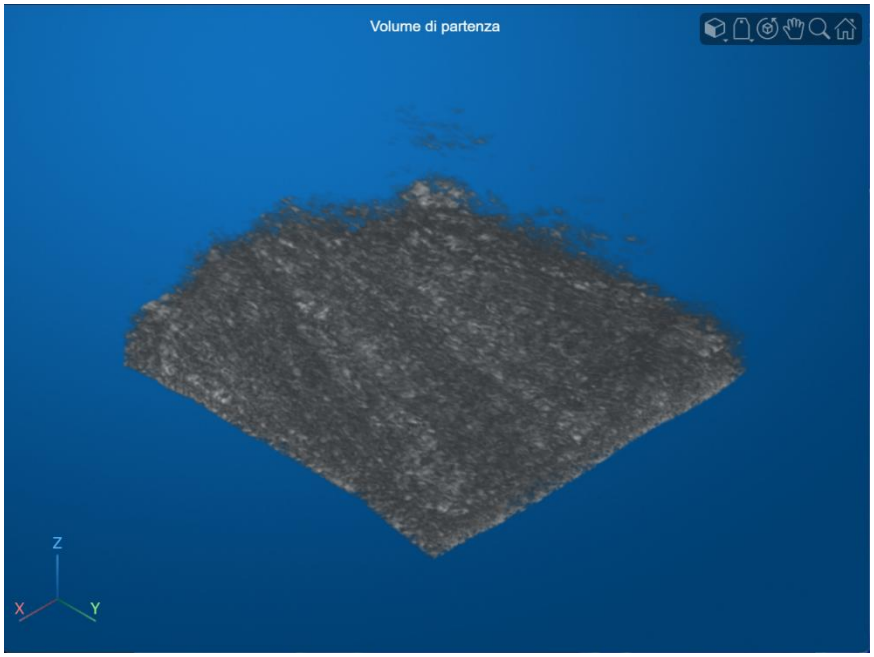



*Figura 2: Rappresentazione grafica della matrice M relativa all'acquisizione "00" dell'utente "Brienza".*

## 2.2. Ridimensionamento

La funzione ***cropMatrice.m*** ridimensiona la matrice M, di dimensioni originali 835×820×320 voxel, estraendone una porzione centrale di 650×650×320 voxel.

La zona centrale del volume contiene le informazioni più rilevanti per l'analisi, poiché qui le vene risultano maggiormente visibili e distinguibili. Al contrario, i bordi del volume presentano un'elevata quantità di "rumore" e artefatti indesiderati che complicano il processo di feature extraction; questi disturbi, essendo troppo attaccati alle vene, sono difficili da separare senza compromettere la qualità d'insieme del pattern venoso. Il cropping centrale permette di isolare la parte interna della matrice, riducendo l'influenza del rumore periferico e preservando dettagli rilevanti per l'estrazione delle feature, senza "sporco" eccessivo.

Inoltre, la riduzione della matrice M a una risoluzione più contenuta riduce significativamente il numero di voxel da elaborare, con un conseguente risparmio nei tempi di calcolo. Questo è particolarmente vantaggioso nelle successive fasi di estrazione delle feature, che altrimenti richiederebbero un elevato carico computazionale per l'analisi dell'intero volume originario.

Di seguito è possibile osservare sempre la precedente matrice M ridimensionata come appena descritto:

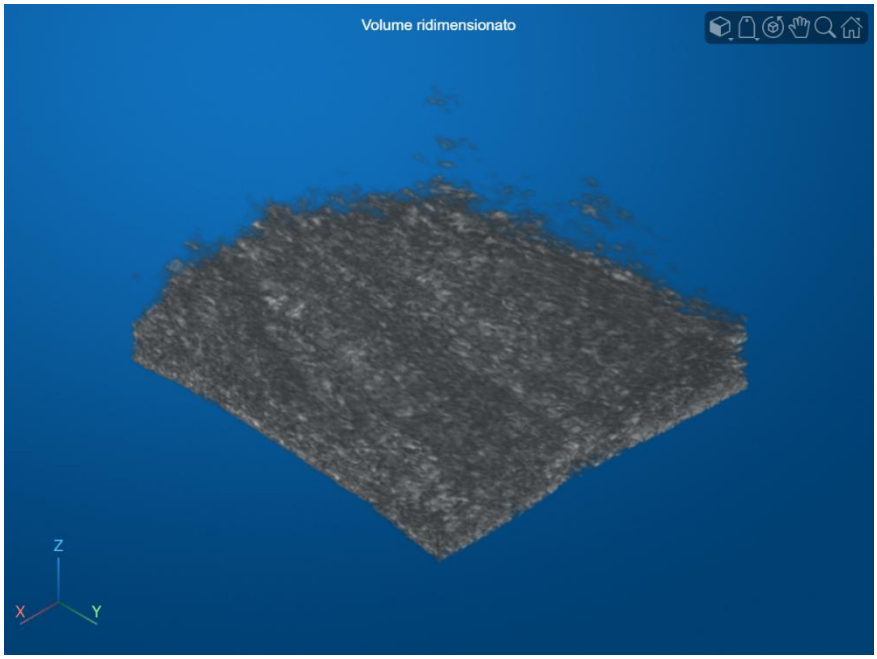



*Figura 3: Rappresentazione grafica della solita matrice M dopo il ridimensionamento (ovvero Mc).*

## 2.3. Estrapolazione

La funzione ***estrapolaVolumeVene.m*** ha lo scopo di isolare, a partire dal volume $M$, la porzione realmente utile del palmo in cui ricercare il pattern venoso, eliminando le componenti non informative (acqua/rumore e regioni nere) e producendo anche maschere di supporto per i passaggi successivi. Per prima cosa stima la superficie del palmo usando una soglia di intensità ($tresh = 64$): costruisce un volume di indici di profondità coerente con la convenzione dell'acquisizione (tramite flip), azzera tali indici dove l'intensità è sotto soglia e, prendendo il massimo lungo la profondità, ottiene una mappa 2D che per ogni coordinata (x, y) identifica il voxel sopra-soglia più superficiale, cioè la quota della superficie del palmo. Questa superficie viene poi regolarizzata con un filtro passa-basso (media 20×20) per ridurre rumore e discontinuità.

Successivamente la funzione deve trasformare due quantità che hanno un significato "fisico" (espresse in millimetri) in quantità "discrete" compatibili con un volume campionato a slice. Il volume 3D infatti non è continuo: lungo l'asse di profondità (z) è composto da un certo numero di piani, e la distanza reale tra due slice consecutive non è 1 mm "per definizione", ma dipende da come è stata fatta l'acquisizione. Questa informazione è contenuta nel vettore Z (anch'esso presente come visto prima nel solito file mat), che rappresenta le profondità (in mm) associate alle slice: in pratica $Z(k)$ indica a quale profondità reale corrisponde la k-esima slice. Di conseguenza, la differenza $Z(2) - Z(1)$ fornisce il passo di campionamento lungo z, cioè quanti millimetri separano due slice adiacenti.

A questo punto la funzione prende due parametri: $depth$ e $thick$. Il parametro $depth$ (0.2 mm) indica di quanto si vuole spostare la superficie stimata verso l'interno (o comunque lungo la profondità) per scegliere una quota di riferimento più adatta all'analisi: non si lavora esattamente sulla superficie stimata, ma su un piano leggermente traslato, perché la zona immediatamente superficiale può essere più rumorosa o meno informativa. Il parametro $thick$ (0.2 mm) rappresenta invece lo "spessore" della regione che si vuole considerare attorno a quella quota, cioè una fascia in profondità che non è un singolo piano, ma un intervallo di slice.

Per poter applicare queste scelte in un volume indicizzato, la funzione converte i millimetri in numero di slice. Lo fa dividendo ciascuna distanza per il passo $dz = Z(2) - Z(1)$ e arrotondando: $depth\_ind = round(depth/dz)$ dice quante slice corrispondono a 0.2 mm, mentre $thick\_ind = round(thick/dz)$ dice quante slice servono per coprire 0.2 mm di spessore. Una volta ottenuti questi indici, la funzione può definire in modo coerente la "fascia" attorno alla superficie stimata: prima sposta la superficie filtrata di $depth\_ind$ slice per posizionarsi alla profondità desiderata, poi usa $thick\_ind$ per stabilire quanto ampia deve essere la zona di transizione (la rampa) e quindi quali voxel devono essere considerati sopra la fascia (da scartare), dentro la fascia (transizione graduale) e sotto la fascia (da mantenere). In questo modo la selezione della regione in profondità rimane espressa in millimetri (quindi interpretabile fisicamente), ma viene applicata correttamente sui dati discreti del volume 3D.

A questo punto costruisce una maschera tridimensionale A con valori tra 0 e 255: i voxel al di sopra della fascia vengono posti a 0 (da scartare), quelli al di sotto a 255 (da mantenere) e, se abilitata la trasparenza, viene applicata una rampa intermedia per ottenere una transizione morbida nello spessore considerato. Da A ricava la mascheraAcqua (regioni sotto-soglia) e azzera tali voxel nel volume, rimuovendo quindi acqua e rumore.

Il volume e la maschera vengono poi ri-orientati per essere compatibili con la convenzione usata dal resto della pipeline (inversione dell'asse di profondità e permutazione delle dimensioni per ottenere l'ordine [y, x, z]). Di seguito è possibile apprezzare una visualizzazione del volume ripulito dall'acqua.

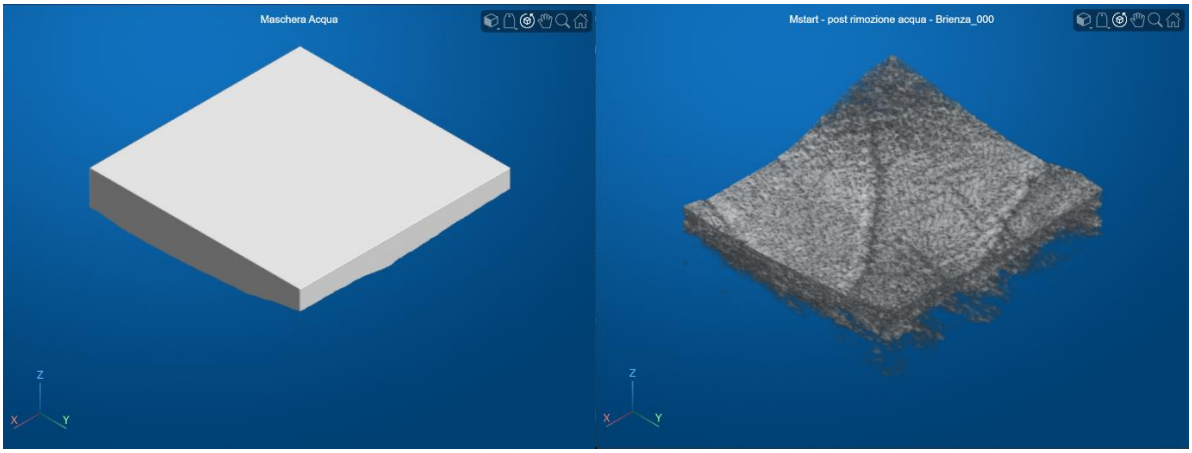
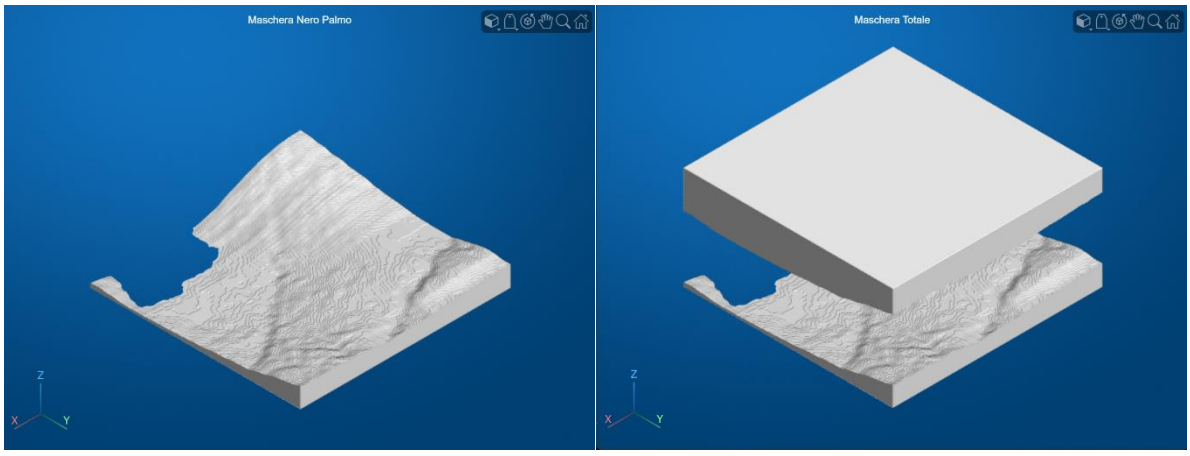



*Figura 4.a: Maschera volta all'eliminazione dell'acqua; Figura 4.b: Rappresentazione grafica della matrice Mc dopo la rimozione dell'acqua (ovvero Mstart).*

In seguito, tramite la funzione ***calcolaMaschere.m***, vengono calcolate le maschere delle regioni nere e viene ricavato *volumePalmo*, cioè il volume che rappresenta la regione del palmo effettivamente utilizzabile. Combinando la maschera del nero del palmo con la maschera dell'acqua, la funzione produce anche *Mnp*, una versione del volume in cui sono stati rimossi sia acqua/rumore sia le porzioni nere del palmo. Di seguito è possibile apprezzare una visualizzazione del volume ulteriormente ripulito.



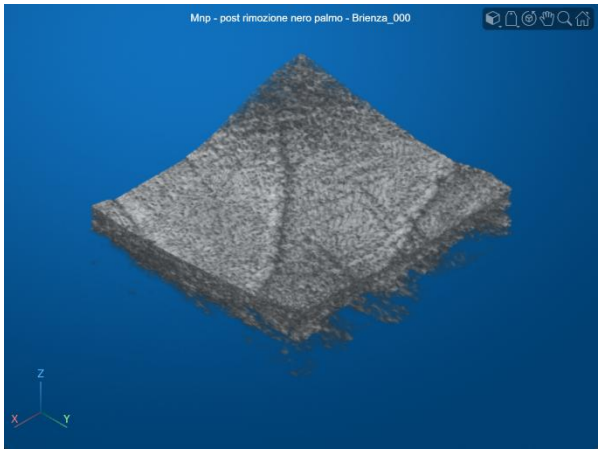




*Figura 5.a: Maschera volta all'eliminazione del nero del palmo; Figura 5.b: Maschera totale che alla fine è stata applicata sul volume di partenza; Figura 5.c: Rappresentazione grafica della matrice Mnp dopo la rimozione del nero del palmo.*

Infine, per ogni coordinata (x, y), la funzione calcola l'indice della prima slice lungo z in cui $volumePalmo$ risulta non nullo, ottenendo una mappa della "quota di inizio palmo" ($indiciPalmo$). Sottraendo un offset fisso ($offsetPelle = 10$) restituisce indiciPalmoNoPelle, che rappresenta una stima dell'inizio del palmo al netto dello strato più superficiale (pelle), utile per concentrare l'analisi su profondità più informative per il pattern venoso.

## 2.4. Binarizzazione

La funzione ***effettuaBinarizzazione.m*** realizza l'intera pipeline di **binarizzazione 3D** finalizzata all'estrazione delle **strutture venose** dal volume del palmo. L'input principale è il volume pre-processato ***Mnp*** (con intensità in 0–255), accompagnato dalle maschere di esclusione (mascheraAcqua, mascheraNeroTotale) e dalla mappa di superficie del palmo senza pelle indiciPalmoNoPelle, utilizzata per riferire le operazioni alla geometria del palmo lungo la profondità. In uscita produce: (i) ***volBinFinal***, ossia il volume binario finale delle vene, e (ii) ***vecProcessed***, un vettore "smussato" che descrive, per ciascun piano y, fino a quale profondità conviene binarizzare. Tale funzione si compone di diversi passi:

1) Calcolo del vettore fine binarizzazione;
2) Binarizzazione iniziale e pulizia morfologica;
3) Stima distanze vene–palmo e stima dei diametri;
4) Stima automatica della soglia iniziale;
5) Binarizzazione finale (incrementale e vincolata);
6) Salvataggio su disco.

Ognuno dei passi precedenti, essendo tutti piuttosto complicati, è effettuato da una o più funzioni dedicate.

### 2.4.1. Calcolo del vettore fine binarizzazione

La funzione ***calcolaVecFineBin.m*** costruisce un vettore ***vecFineBin*** (lungo $yDim$) che, per ogni slice lungo y del volume, stima un valore ottimale del parametro **fine** usato poi nella binarizzazione dei piani XZ: in pratica indica fino a che profondità sotto la superficie del palmo conviene "scendere" quando si binarizza quel piano.

Per prima cosa legge le dimensioni del volume assumendo la convenzione (y, x, z), crea un volume a contrasto invertito $Minv = 255 - Mn$ (così strutture originariamente scure diventano ad alta intensità) e azzera tutte le zone non utili usando $Minv(mascheraNeroTotale) = 0$. Fissa inoltre $inizio = 1$, cioè il limite inferiore dell'intervallo di profondità considerato nella binarizzazione.

La stima di fine avviene poi con una scansione controllata: dalla maschera acqua ricava, per ogni coppia (y, x), l'indice z del primo voxel marcato come acqua tramite $[\sim, id\_mascheraAcqua] = max(mascheraAcqua \sim = 0, [], 3)$ e impone un limite globale conservativo $fineIterazione = min(id\_mascheraAcqua(:)) - 1$, cioè non prova mai valori di fine oltre una profondità che potrebbe già ricadere in acqua in qualche punto del volume. A questo punto, per ogni y, estrae il piano XZ del volume invertito come $pianoXZ = squeeze(Minv(y, :, :))'$ (porta il piano nella forma attesa dalle funzioni successive), prende il profilo della superficie "palmo senza pelle" $idPalmoNoPellepiano = indiciPalmoNoPelle(y, :)$ e costruisce un vettore vecCC lungo $fineIterazione$. Per ogni candidato binarizza il piano chiamando ***binPianoSingolo.m***, converte la maschera logica risultante in un'immagine binaria uint8 a 0/255 (formato richiesto dalla funzione di conteggio) e calcola quante componenti connesse sono presenti con $[numCC, \sim] = contaCC(pianoFinaleBin, y)$, salvando $numCC$ in $vecCC(fine)$. Terminata la scansione, cerca i picchi di vecCC con $findpeaks(\ldots, 'MinPeakProminence', 1)$ e sceglie l'indice ottimo del piano in modo deterministico: se i picchi sono almeno due usa la media dei primi due, se ce n'è uno solo usa il primo, se non ce ne sono assegna NaN. Quel valore viene salvato in vecFineBin(y). Finito il ciclo su tutte le y, crea la cartella di output se manca e salva vecFineBin su disco solo se contiene almeno un valore diverso da zero altrimenti evita di salvare un risultato vuoto.

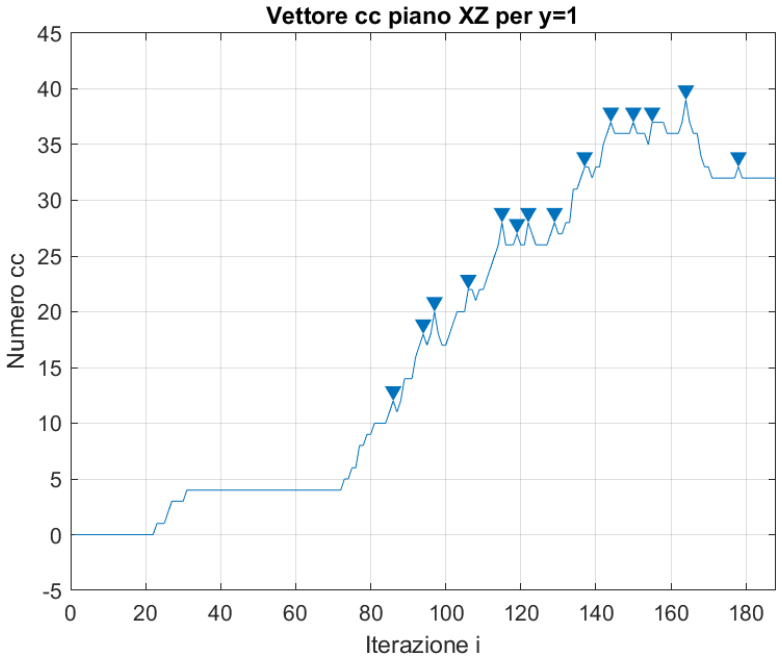



*Figura 6: Plot del numero delle cc per ogni iterazione di un piano XZ.*

La scelta di limitarsi ai primi due picchi è del tutto arbitraria e potrebbe rivelarsi non ottima.

Una volta completati tutti i piani il vettore ricavato viene processato tramite la funzione ***processaVettore.m*** la quale restituisce il vettore $vecProcessed$.

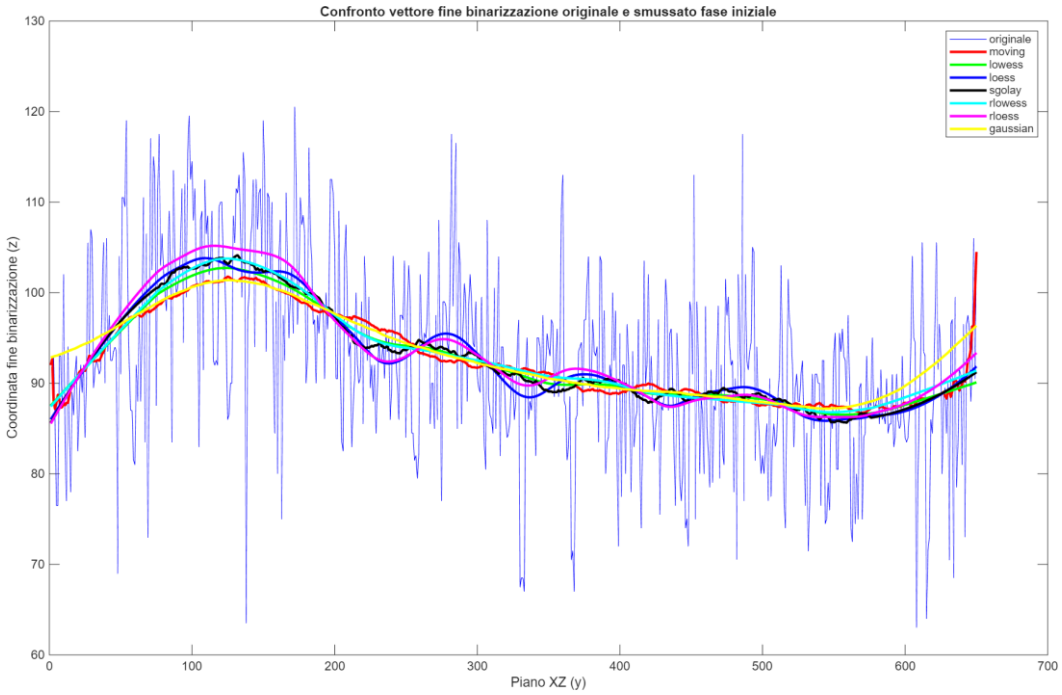



*Figura 7: Fine binarizzazione calcolata per ogni piano XZ del volume.*

Nella Figura 7 è possibile vedere il valore di fine ottimo calcolato per ogni piano XZ il quale rappresenterà l'andamento della binarizzazione lungo la dimensione Z e quindi sarà in grado di ricavare sia le vene più pendenti che quelle costanti. Per rendere la caratterizzazione delle vene più omogenea viene applicata al vettore in questione una procedura di smoothing. Lo smussamento va a regolare i picchi non dando peso a possibili outlier ma conservando l'andamento generale delle vene profonde. Sono stati provati diversi operatori di smoothing ma alla fine si è visto che l'operatore rlowess funziona meglio per questo scopo (tratto turchese).

Da questo vettore possiamo intuire come l'utente in questione abbia nella zona del BSCAN che va da 0 a 200 la vena principale ad una profondità che parte nell'intorno di 90 per poi scendere in profondità fino a 100 e infine risalire più verso la superficie da 200 in poi.

## 2.4.2. Binarizzazione iniziale e pulizia morfologica

La funzione ***binarizzaVolume.m*** esegue la binarizzazione 3D del volume lavorando piano per piano, sui piani XZ (uno per ogni indice y). Nel contesto di questo step "iniziale" viene invocata con $sogliaIniziale = 0$ e $sogliaFinale = 0$, condizione che attiva la modalità di binarizzazione preliminare (non incrementale) pensata per ottenere una prima estrazione "grezza ma affidabile" delle vene.

All'inizio la funzione prepara un volume di intensità adatto alla soglia: calcola $Minv = 255 - Mn$, cioè inverte il contrasto, e impone a zero tutte le regioni non utilizzabili tramite $mascheraNeroTotale$. L'inversione è importante perché rende "alte" (quindi facilmente sogliabili) le strutture che nell'immagine originale sono più scure; la maschera invece elimina a monte zone nere/acqua o comunque non affidabili, impedendo che generino falsi positivi. Viene poi allocato un volume binario finale $MatFinale$ (tutto falso) delle stesse dimensioni del volume originale (convenzione y, x, z), che verrà riempito progressivamente.

La binarizzazione vera e propria è eseguita indipendentemente per ogni y (in parallelo). Per ciascun piano, viene letto il parametro $fine = vecFine(y)$, che rappresenta fino a quante slice di profondità sotto la superficie conviene analizzare in quel piano. A questo punto entra in gioco la mappa $indiciPalmoNoPelle(y, x)$, cioè l'indice z della superficie del palmo (senza pelle) per ogni colonna x: la soglia non viene applicata su una profondità assoluta fissa, ma su una fascia di profondità relativa alla superficie locale. In pratica, per ogni profondità relativa compresa tra 1 e fine, e per ogni x, si campiona il voxel situato a quota $z = indiciPalmoNoPelle(y, x) - offset$ (se l'indice resta dentro il volume) e lo si marca come appartenente a vena se l'intensità invertita supera una soglia molto alta. In questa fase la soglia usata è 250 su Minv, quindi vengono selezionati solo voxel "quasi saturi" dopo inversione: equivalgono, nell'immagine originale, a intensità molto basse (strutture molto scure), scelta coerente con una binarizzazione iniziale volutamente conservativa.

Il risultato ottenuto con questa prima soglia è poi raffinato con un passaggio aggiuntivo (***binOffsetVena.m***), che serve a "ritagliare" e rendere più coerente la regione lungo z in cui mantenere le attivazioni. In particolare, sul piano XZ binarizzato vengono calcolate le componenti connesse 2D e i relativi centroidi; viene individuata la componente con centroide più "superficiale" (minimo in z) e, a partire da quella quota, la maschera viene limitata/propagata mantenendo solo le attivazioni entro un offset fisso di 60 slice verso profondità maggiori. Operativamente, questo passaggio tende a scartare attivazioni troppo superficiali o isolate e a concentrare la maschera nella fascia in cui la vena risulta più plausibile e consistente nel piano. Il piano così ottenuto viene poi riportato nel volume 3D finale (riallineando correttamente gli assi del piano rispetto alla convenzione y, x, z). L'output di questa modalità è quindi un volume binario preliminare **volBin** che rappresenta una prima estrazione delle vene, costruita (i) rispettando la geometria del palmo tramite indiciPalmoNoPelle, (ii) limitando la profondità con vecFine, e (iii) usando una soglia molto selettiva per ridurre rumore e falsi positivi. Il risultato ottenuto è visibile nell'immagine sottostante:

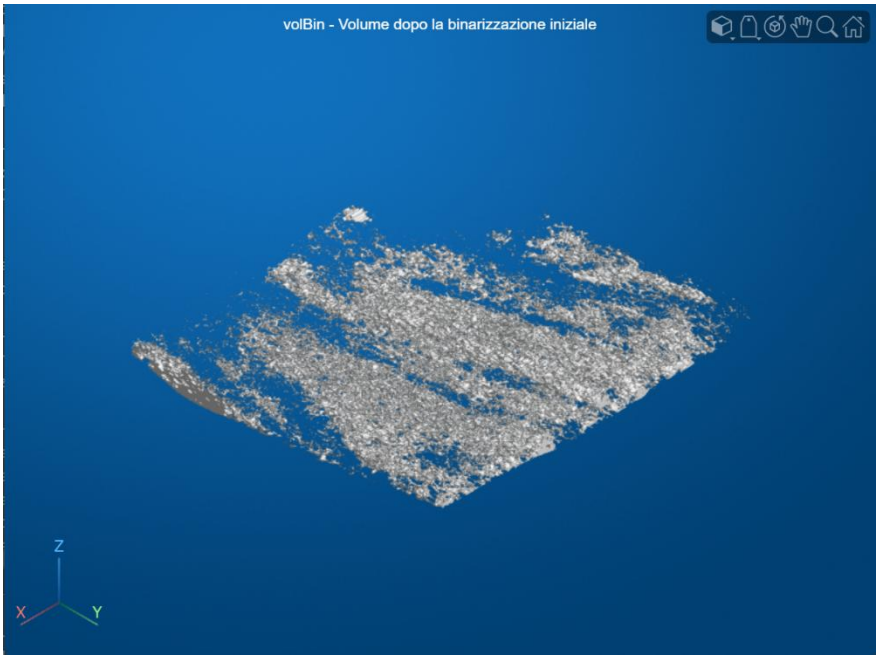



*Figura 8: Rappresentazione grafica della matrice volBin dopo una prima binarizzazione iniziale.*

La funzione ***separaStrutture.m*** prende in ingresso il volume binario prodotto dalla binarizzazione iniziale e lo trasforma in un volume "pulito" e più stabile, eliminando rumore, frammenti e connessioni spurie. Il primo intervento è un filtraggio per dimensione tramite *bwareaopen* con soglia 500 voxel: tutte le componenti connesse più piccole vengono rimosse, perché tipicamente riconducibili a rumore o attivazioni isolate non venose. Dopo questo step il volume viene portato in formato 0/255 (valori utili sia per debug sia per alcune funzioni successive), mantenendo però la natura binaria dell'informazione.

Successivamente viene applicata un'apertura morfologica 3D (*imopen*) con elemento strutturante sferico di raggio 1. Questa operazione (erosione seguita da dilatazione) ha lo scopo di eliminare piccole sporgenze, interrompere ponti sottili e separare strutture che risultano connesse solo per contatti minimali, migliorando la separabilità degli oggetti effettivamente presenti. A valle dell'apertura, la funzione applica anche uno smoothing gaussiano 3D con sigma 1 e padding "circular": l'effetto è quello di ammorbidire i bordi e colmare micro-discontinuità, rendendo la successiva ri-binarizzazione meno sensibile a irregolarità locali.

Dopo lo smoothing, il volume viene nuovamente binarizzato con una soglia manuale pari a 100 (cioè si torna a una rappresentazione netta dopo aver reso le strutture più regolari). Infine viene eseguita una seconda pulizia più severa: un ulteriore *bwareaopen* mantiene soltanto le componenti connesse con volume almeno 2000 voxel, eliminando i residui rimasti dopo la fase di smoothing e ri-soglia. In termini funzionali, separaStrutture produce dunque un volume binario molto più consistente e robusto, che viene poi utilizzato negli step successivi (ad esempio stima distanza

vene–palmo e diametri), dove è fondamentale che le componenti siano sufficientemente grandi e coerenti per estrarre misure geometriche affidabili. Il risultato ottenuto è mostrato nella figura seguente:


*Figura 9: Rappresentazione grafica della matrice volumeSeparato dopo una prima serie di operazioni morfologiche atte a separare le strutture dal rumore.*

### 2.4.3. Stima distanze vene–palmo e stima dei diametri

Dopo la binarizzazione iniziale e la pulizia morfologica (che producono un volume di vene già abbastanza coerente), la funzione ***calcolaMinDistVenePalmo.m*** ha il compito di ricavare, per ogni piano XZ (cioè per ogni indice y), una misura geometrica fondamentale: la distanza minima tra le vene segmentate e il palmo lungo la direzione z. In parallelo tenta anche di stimare, sullo stesso piano, un diametro plausibile della vena (quando possibile). Queste due informazioni diventano poi vincoli essenziali per la binarizzazione "finale": la distanza aiuta a capire fino a che profondità conviene cercare le vene, mentre il diametro serve come parametro geometrico per rendere più robusta l'estrazione.

La funzione inizia con un taglio "grossolano" dei bordi: azzera manualmente due fasce laterali lungo x (da 1 a 50 e da 600 a 650) sia nel volume vene sia nel volume palmo. Questo step è un filtro empirico per eliminare regioni esterne che, per caratteristiche del sensore o della porzione acquisita, risultano frequentemente rumorose o poco affidabili. Subito dopo applica un ulteriore taglio mediante ***tagliaBordi.m***, che rimuove porzioni agli angoli (in modo più "geometrico" e uniforme su tutte le slice), riducendo ancora la probabilità che artefatti periferici influenzino misure di distanza e diametro. A scopo di controllo visivo viene anche costruito un volume unito (OR logico tra palmo e vene) e mostrato con

***graficoVolshow***, così da verificare che le vene risultino effettivamente "sotto" il palmo e che i tagli non abbiano eliminato regioni di interesse.

Il calcolo vero e proprio avviene poi piano per piano (loop parallelo su y). Per ogni y, la funzione estrae il piano XZ delle vene e quello del palmo, e applica al piano vene una piccola pulizia 2D: un'erosione con elemento strutturante sferico di raggio 1 e un riempimento dei buchi ($imfill$). Questo serve a stabilizzare le regioni venose nel singolo piano, eliminando pixel isolati e rendendo le strutture più compatte, così che il successivo conteggio/filtraggio per componenti sia più affidabile. Successivamente calcola le componenti connesse del piano venoso e seleziona solo quelle con area compresa tra 20 e 2500 pixel: l'obiettivo è scartare sia rumore (oggetti troppo piccoli) sia regioni troppo grandi (che difficilmente rappresentano una singola vena plausibile o che potrebbero essere aggregazioni spurie). Il risultato di questo filtraggio è una maschera di "vene valide" del piano.

A questo punto la funzione prova anche a stimare un diametro tramite ***calcolaDiametroVena.m***, usando la maschera delle componenti valide e la geometria del palmo nel piano. Il diametro può risultare $NaN$ se non ci sono condizioni sufficientemente affidabili per stimarlo (ad esempio assenza di strutture adeguate o geometria non coerente). In parallelo viene calcolata la distanza vene↔palmo: per ogni colonna x del piano, si cerca nel palmo l'ultimo voxel non nullo lungo z (interpretato come "quota più profonda" del palmo in quella colonna) e, nella maschera vene valide, si cercano gli ultimi 3 voxel non nulli lungo z (cioè la parte più "profonda" della vena, imponendo però un requisito minimo di consistenza per evitare falsi positivi dovuti a un singolo voxel isolato). Solo se in quella colonna sono presenti almeno 3 voxel venosi, la distanza viene considerata misurabile e viene calcolata come $indicePalmo - indiceVena$ (distanza in voxel lungo z). Ripetendo questa operazione su tutte le colonne x, la funzione ricava un vettore di distanze per il piano e ne salva la minima: questa scelta identifica, per quel piano y, il caso in cui la vena risulta più "vicina" al palmo (misura conservativa utile per limitare la profondità di ricerca nelle fasi successive). Se per quel piano esiste anche un diametro valido, la funzione memorizza sia il diametro sia la distanza associata (che, in questa implementazione, coincide con la minima distanza del piano). In uscita produce quindi: ***vettoreDistanze*** (una distanza minima per ogni y) e ***matriceDiametri*** ($2\times yDim$, con diametri stimati e distanze associate).

La funzione ***processaVettore.m*** viene usata subito dopo per trasformare un vettore grezzo calcolato per piani (come **vecDist**) in un profilo **continuo, stabile e utilizzabile** nelle fasi successive. In pratica, questi vettori possono contenere $NaN$ (piani in cui la misura non è stimabile) e oscillazioni locali dovute a rumore o a segmentazioni irregolari: prima di usarli come vincoli nella binarizzazione finale, conviene renderli "regolari" e senza buchi.

Il primo passaggio è la gestione dei valori mancanti: la funzione interpola linearmente i $NaN$ usando solo i campioni validi. Poiché l'interpolazione può lasciare $NaN$ agli estremi (tipicamente all'inizio, se le prime slice non hanno valori), viene gestito esplicitamente il caso in cui manchino i primi campioni: questi vengono riempiti copiando il primo valore disponibile, così da evitare un inizio indefinito del profilo. Una volta ottenuto un vettore completo, la funzione applica una fase di smoothing piuttosto forte (finestra pari a **150** campioni), calcolando diverse varianti di levigatura (media mobile, lowess/loess, Savitzky–Golay, Gaussian, e versioni robuste). Anche se ne calcola molte per confronto, l'uscita effettivamente selezionata è la versione robusta **rlowess**, scelta perché meno sensibile a picchi e outlier rispetto a una regressione locale standard. Infine, per garantire coerenza dimensionale con il numero di piani del volume, la funzione forza la lunghezza del risultato a $yDim$: se il vettore risultasse più corto (ad esempio per effetti di shape/colonne), aggiunge campioni in coda replicando l'ultimo valore disponibile.

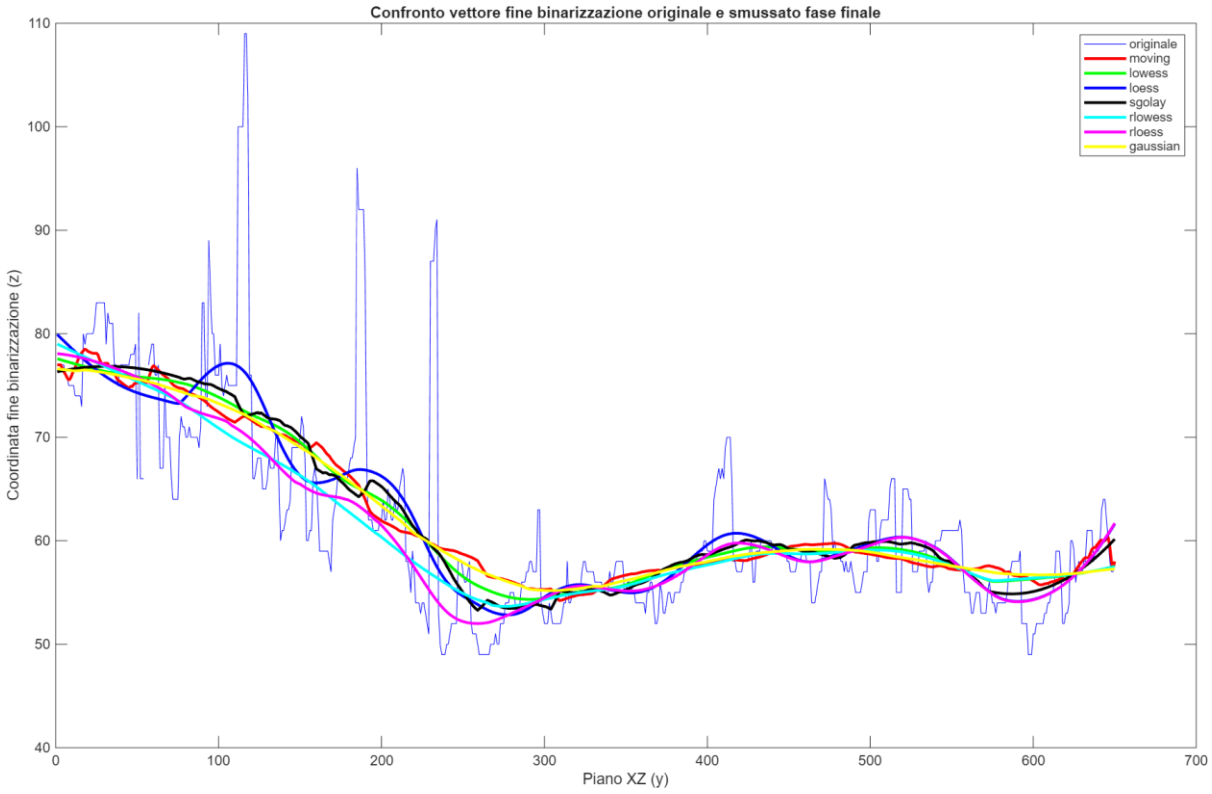



*Figura 10: Distanza palmo-vena calcolata per ogni piano XZ del volume.*

## 2.4.4. Stima automatica della soglia iniziale

Dopo aver ottenuto un volume venoso "pulito" e, soprattutto, un profilo geometrico affidabile della distanza vene–palmo ($vecDistProcessed$), la funzione ***calcolaSogliaIniziale.m*** serve a determinare in modo automatico una soglia di intensità iniziale da cui avviare la binarizzazione finale (quella incrementale). L'idea è evitare una scelta manuale e rendere la procedura ripetibile: la soglia viene

selezionata osservando come cambia la complessità del volume binarizzato al variare della soglia, misurata tramite il numero di componenti connesse "significative" (cioè non troppo piccole).

Operativamente la funzione esegue uno sweep di soglie decrescenti da 250 a 210 (step −1). Per ogni soglia candidata binarizza l'intero volume con ***binIncrementale***, imponendo però un vincolo fondamentale: la profondità massima di analisi viene fissata a un valore conservativo $fine = min(vecDistProcessed)$, cioè la minima distanza stimata tra vene e palmo lungo tutte le slice y. In questo modo, durante il confronto tra soglie, la binarizzazione avviene sempre in una fascia sicuramente compatibile con l'ipotesi "vene sotto il palmo", evitando che profondità variabili introducano artefatti nella metrica. Dal volume binarizzato ottenuto a ciascuna soglia, vengono poi calcolate le componenti connesse 3D e, tramite ***regionprops3***, viene misurato il volume (in voxel) di ogni componente; si considerano "valide" solo quelle con $Volume \geq 2000\ voxel$ e si conta quante sono. Il risultato è quindi un vettore $vecNumCC$ che, per ciascuna soglia provata, contiene il numero di componenti connesse grandi emerse dalla binarizzazione.

A questo punto la scelta della soglia non viene fatta prendendo banalmente il massimo o un valore fisso, ma individuando un punto di transizione nella curva #CC(soglia). Per ridurre l'effetto del rumore dovuto alla discretizzazione, $vecNumCC$ viene smussato con un fitting LOESS (finestra 10) e sulla versione smussata viene calcolata la derivata discreta $dy$, che mette in evidenza dove il numero di componenti cresce rapidamente o cambia regime. La funzione trova prima il massimo globale della curva originale $vecNumCC$ (il punto in cui il numero di CC è massimo), poi cerca i massimi locali della derivata (zone di salita rapida) e i minimi locali della derivata (zone in cui la crescita rallenta/inverte). Viene quindi selezionato: (1) il massimo locale della derivata più vicino al massimo globale della curva e, (2) il minimo locale della derivata più vicino a quel massimo locale. La soglia iniziale finale è la soglia corrispondente a questo minimo locale "successivo", cioè un punto in cui si è appena superata la fase di variazione più brusca: in termini pratici, è un compromesso che tende a evitare sia soglie troppo alte (troppo selettive, poche strutture) sia soglie troppo basse (troppo permissive, frammentazione/rumore), scegliendo un valore coerente con la dinamica osservata sul volume.

Di seguito è possibile osservare i grafici di interesse dei vettori appena discussi:

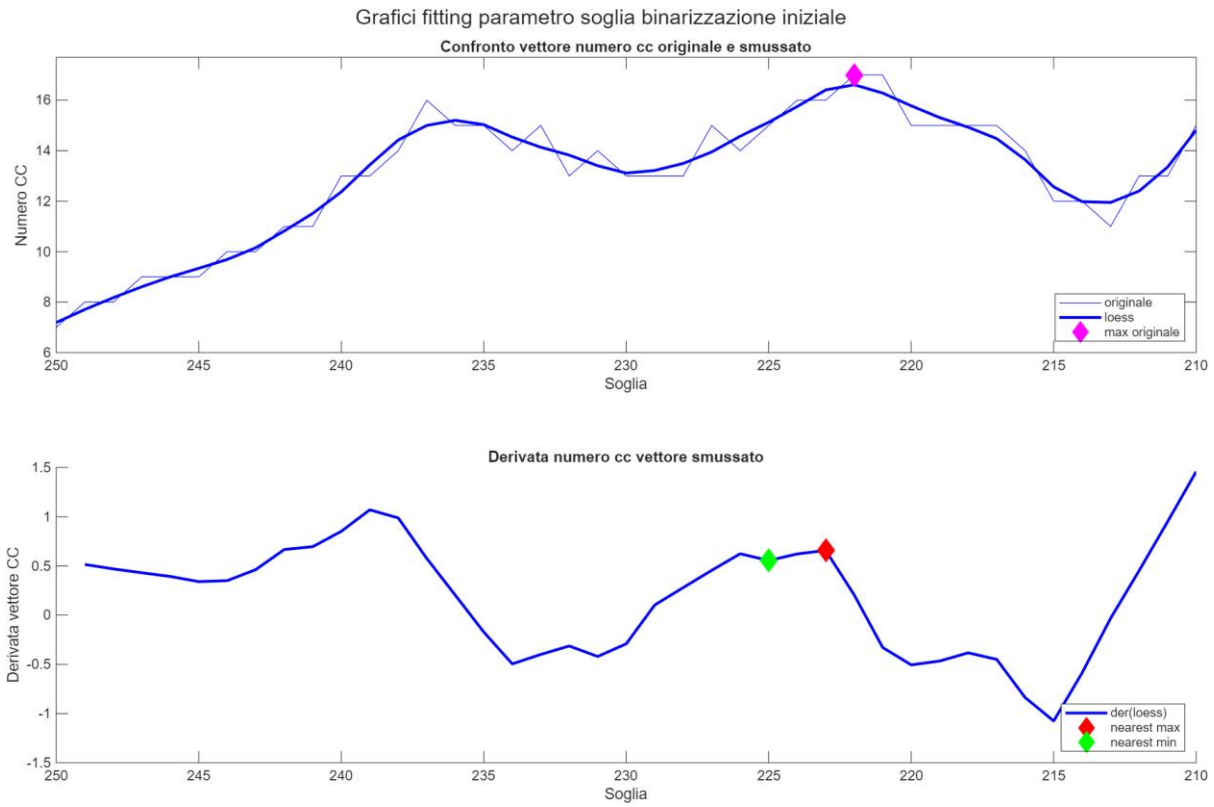



*Figura 11: Plot numero cc in funzione del valore di soglia iniziale e sua derivata.*

## 2.4.5. Binarizzazione finale

Nello step finale di ***effettuaBinarizzazione***, la funzione ***binarizzaVolume.m*** viene richiamata passando $vecDistProcessed$ come vettore $vecFine$, la $matriceDiametri$ stimata in precedenza e le soglie $sogliaIniziale \rightarrow 255$. In questa modalità la binarizzazione non è più "manuale" e fissa, ma diventa incrementale lungo la profondità e viene anche adattata tramite un parametro geometrico globale (la cosiddetta $costanteDiametroVene$), ricavato dai diametri stimati.

La funzione parte come nella fase iniziale: lavora nel riferimento del volume (y, x, z), crea $Minv = 255 - Mn$ per rendere alte le strutture originariamente scure e azzera tutte le regioni non utilizzabili con $mascheraNeroTotale$. Il risultato finale viene accumulato in un volume binario $MatFinale$ inizializzato a false. A questo punto, però, prima di binarizzare, viene stimato un offset di profondità da aggiungere a $vecFine$ per rendere la ricerca più robusta e coerente con la "taglia" delle vene.

Dalla $matriceDiametri$ la funzione legge, per ogni piano y, un diametro stimato ($diametro$) e una distanza associata ($distDiametro$). Non tutti questi diametri vengono accettati: vengono considerati "validi" solo quelli per cui $distDiametro$ è coerente con il valore $vecFine(y)$, misurando una differenza percentuale e imponendo che sia $< 5\%$. Questa regola serve a evitare di usare diametri stimati su piani in cui la

stima geometrica (o la segmentazione) è poco affidabile o non compatibile col profilo di profondità usato per la binarizzazione.

I diametri che superano il controllo vengono raccolti e la costante viene inizialmente posta come media arrotondata dei diametri validi (se non ce ne sono, la costante viene posta a 0). In pratica, questa costante è un parametro globale che "espande" la profondità di ricerca: se le vene hanno mediamente un certo spessore/estensione, la profondità utile non dovrebbe fermarsi esattamente a $vecFine(y)$, ma includere una fascia aggiuntiva.

Se la costante iniziale è > 0, viene poi raffinata automaticamente da ***calcolaCostanteDiametroVene***: qui l'idea è provare più offset (da 0 fino alla costante stimata) e scegliere quello che produce un comportamento più "informativo" nel numero di strutture venose robuste. Per ogni tentativo, la funzione binarizza il volume con ***binIncrementale*** usando come profondità $max(vecFine) + tentativo$ e conta quante componenti connesse 3D hanno volume almeno 3000 voxel (si contano quindi solo strutture grandi, meno compatibili con rumore). Si ottiene una curva "tentativo → numero CC robuste", la si smussa (loess), si guarda la derivata discreta e si scelgono punti di variazione (massimi/minimi locali della derivata): se i diametri validi sono molti (>20) la scelta tende a essere più "spinta" (punto più tardivo), altrimenti più conservativa (punto più precoce); se non esistono punti di variazione, si ripiega sul tentativo che massimizza direttamente il numero di CC robuste. Il risultato è una $costanteDiametroVene$ finale, più stabile rispetto alla sola media.

Definita la costante, la binarizzazione finale viene eseguita in parallelo su tutti i piani y. Per ciascun piano si imposta una profondità effettiva: $fine = vecFine(y) + costanteDiametroVene$, cioè si estende la profondità stimata per quel piano aggiungendo un margine legato al diametro medio delle vene. A questo punto entra in gioco ***binIncrementalePiano***, che è la funzione che realizza la binarizzazione vera e propria sul singolo piano XZ. Il principio è lo stesso già visto in altre parti della pipeline: non si soglia un'intera slice a quota z fissa, ma si lavora in coordinate relative alla superficie del palmo. Per ogni "passo" di profondità virtuale zplane da 1 a fine, viene calcolata per ogni colonna x la posizione assoluta lungo z come: $indice\_z = indiciPalmoNoPelle(y, x) - zplane$. Se $indice\_z$ è valido (>0), viene campionato il voxel corrispondente nel volume invertito Minv (quindi sotto la superficie del palmo) e confrontato con una soglia. La particolarità (che giustifica il nome "incrementale") è che la soglia non è costante, ma varia con la profondità: tra sogliaIniziale e sogliaFinale viene interpolata con un andamento quadratico in funzione di t normalizzato tra 0 e 1 (cioè la soglia cambia lentamente all'inizio e più rapidamente verso la fine dell'intervallo). In questo modo ogni voxel "candidato" viene valutato con un criterio che dipende dalla profondità a cui si trova: all'aumentare di *zplane* la procedura tende a rendere la selezione progressivamente più severa (dato che qui sogliaFinale è 255).

Quando un voxel supera la soglia del suo step di profondità, viene scritto come true nel volume 3D esattamente nella posizione (y, x, indice_z). Ripetendo il processo per tutte le profondità virtuali e per tutte le colonne, il piano finale viene costruito come un insieme di voxel selezionati sotto la superficie del palmo, e ***binarizzaVolume*** lo inserisce in ***MatFinale*** come slice y del volume completo.

Il risultato di questo step è quindi un ***volBinFinal*** che nasce da tre vincoli combinati: (1) vincolo geometrico dato dalla superficie del palmo (***indiciPalmoNoPelle***), (2) vincolo di profondità dato da ***vecFine*** esteso con una costante legata al diametro venoso, e (3) criterio di selezione adattivo lungo la profondità tramite soglia incrementale. Questo volume finale è quello che ***effettuaBinarizzazione*** salva e verrà utilizzato poi per il matching/riconoscimento. Nella figura sottostante è possibile osservare come si presenta il volume dopo la binarizzazione finale:

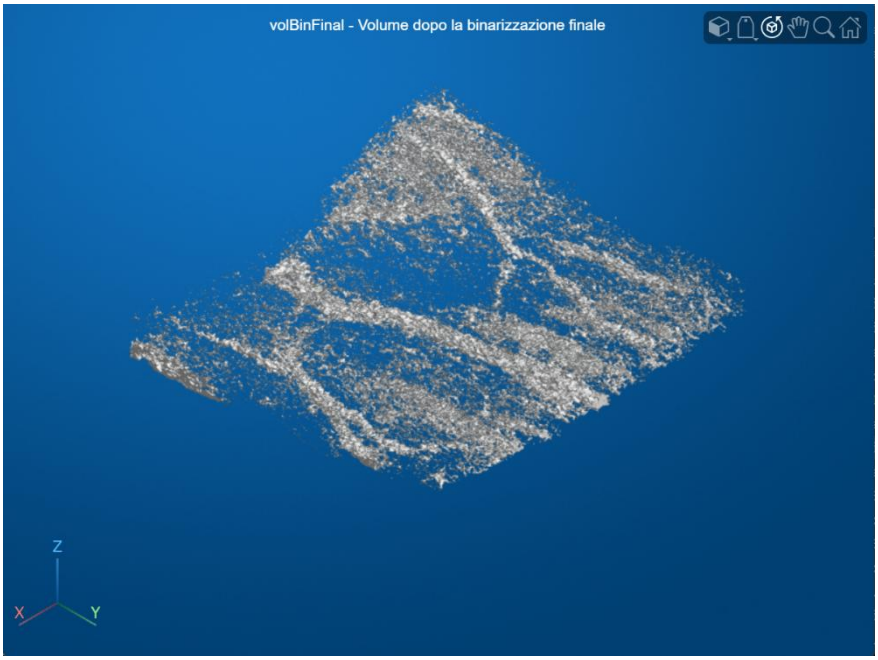



*Figura 12: Rappresentazione grafica della matrice* volBinFinal *dopo la binarizzazione finale.*

## 2.5. Isolamento del pattern venoso

La funzione ***isolaPatternVenoso.m*** ha l'obiettivo di "ripulire" e isolare il pattern venoso a partire da un volume 3D già binarizzato, nel quale le strutture vascolari risultano presenti insieme a componenti spurie dovute a rumore, artefatti di acquisizione o errori di segmentazione. L'output è un volume filtrato, in cui si preservano le strutture più consistenti e coerenti con una geometria venosa, mentre vengono eliminate le componenti di piccole dimensioni e le irregolarità morfologiche.

Il filtraggio del volume avviene tramite una sequenza di operazioni morfologiche e di analisi delle componenti connesse, strutturata in tre passaggi principali.

In primo luogo, viene applicata una rimozione delle componenti connesse piccole mediante *bwareaopen*, eliminando tutti gli oggetti con volume inferiore a una soglia fissata (500 voxel). La scelta di una connettività 3D pari a 6 (adiacenza per facce) rende questo passaggio più restrittivo nella definizione di "connessione" tra voxel: due voxel appartengono alla stessa componente solo se collegati attraverso contatti di faccia, limitando l'unione artificiale di regioni che si toccano solo per spigoli o vertici. In questa fase si ottiene quindi un primo volume "ripulito" dalle strutture minute, tipicamente non compatibili con un albero venoso reale. Successivamente, per ragioni di compatibilità con la visualizzazione e per mantenere coerenza con altri step della pipeline, il volume viene convertito in formato uint16 e i voxel attivi vengono rimappati a intensità 255 (sfondo a 0). Questa scelta non modifica l'informazione binaria sostanziale (presenza/assenza), ma rende più immediata l'ispezione tramite strumenti di rendering volumetrico e funzioni di diagnostica.


*Figura 13: Rappresentazione grafica della matrice volumeCleaned dopo il filtraggio delle componenti connesse piccole.*

Il secondo passaggio applica un'operazione di apertura morfologica 3D ($imopen$), utilizzando un elemento strutturante sferico di raggio 1 voxel ($strel('sphere', 1)$). L'apertura è definita come erosione seguita da dilatazione e viene impiegata per regolarizzare la geometria delle strutture segmentate: l'erosione rimuove piccole asperità, sporgenze sottili e connessioni deboli, mentre la dilatazione ricostruisce le regioni sopravvissute all'erosione mantenendone la forma complessiva. In termini pratici, questa operazione tende a eliminare "filamenti" isolati e irregolarità locali che spesso rappresentano rumore o segmentazioni instabili, migliorando la compattezza delle strutture ritenute affidabili.


*Figura 14: Rappresentazione grafica della matrice volumeOpen dopo l'apertura morfologica 3D.*

Infine, nel terzo passaggio, la funzione esegue un secondo filtraggio delle componenti connesse, più severo in termini di soglia dimensionale: vengono rimossi tutti gli oggetti con meno di 1000 voxel. Diversamente dal primo passaggio, qui viene adottata una connettività 26 (adiacenza per facce, spigoli e vertici), più permissiva nel collegare voxel e quindi più adatta a descrivere strutture sottili e ramificate come le vene, che possono presentare collegamenti diagonali nello spazio discreto. La combinazione "apertura + soglia più alta + connettività 26" ha un effetto selettivo: dopo aver eliminato asperità e connessioni fragili, si conservano prevalentemente strutture volumetricamente significative e spazialmente continue, più plausibili come rete venosa. Il volume risultante viene nuovamente convertito in uint16 e rimappato a 255 per i voxel attivi.

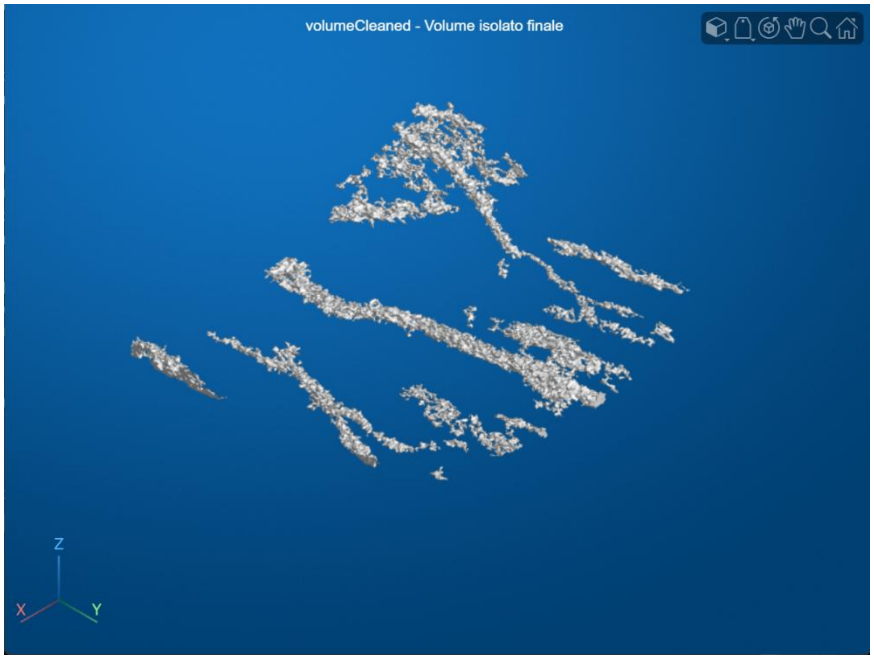



*Figura 15: Rappresentazione grafica della matrice volumeCleaned dopo il filtraggio finale.*

A valle del processo, $volIsolato$ viene salvato su disco (nella cartella degli step intermedi) solo se non risulta vuoto, verificando la presenza di voxel non nulli tramite $nnz$. Questa condizione evita di memorizzare risultati inutili in caso di fallimento della segmentazione o di parametri troppo restrittivi.

Nel complesso, $isolaPatternVenoso$ realizza una post-elaborazione morfologica 3D finalizzata a migliorare la qualità del volume binarizzato: riduce il rumore, regolarizza le forme e impone un criterio dimensionale e topologico (connettività) per mantenere esclusivamente le strutture più coerenti con il pattern venoso atteso, producendo un volume finale adatto a visualizzazione, confronto e successive fasi di matching o analisi quantitativa.

## 2.6. Ispessimento del pattern venoso

La fase di ispessimento ha lo scopo di rendere il pattern venoso binarizzato più robusto e continuo, riducendo la presenza di filamenti spuri e colmando piccole discontinuità che possono compromettere le fasi successive (visualizzazione, confronto o matching). Questo obiettivo viene raggiunto dalla funzione ***inspessimento.m***, che combina un'operazione morfologica 3D con uno smoothing gaussiano volumetrico e una binarizzazione finale a soglia; quando la soglia non è fornita manualmente, essa viene stimata automaticamente tramite la funzione di supporto ***calcolaSogliaBinGauss.m***, basata sull'analisi della variazione del numero di componenti connesse al variare della soglia.

Operativamente, $inspessimento$ adotta innanzitutto una erosione morfologica 3D con elemento strutturante sferico di raggio 1 voxel ($strel('sphere', 1)$). Questa scelta è funzionale a "potare" irregolarità sottili e micro-componenti rumorose: l'erosione riduce localmente lo spessore delle strutture e tende a eliminare porzioni debolmente supportate (rami troppo sottili o isolati), così che le fasi successive ricostruiscano prevalentemente le regioni venose più consistenti.

Dopo l'erosione, il volume viene convertito in un dominio di intensità: il binario/logico viene trasformato in uint8 e i voxel attivi vengono rimappati a 255 (sfondo a 0). Tale passaggio non aggiunge informazione, ma è essenziale per dare significato numerico al filtraggio successivo. Viene quindi applicato un filtro gaussiano 3D ($imgaussfilt3$) con deviazione standard pari a 1, che regolarizza le strutture e introduce un effetto di "riconnessione" di piccoli gap: voxel venosi separati da discontinuità minime possono tornare a costituire una regione più continua grazie alla diffusione locale dell'intensità.

L'ispessimento vero e proprio non è ottenuto tramite una dilatazione morfologica esplicita, ma emerge dall'accoppiata "smoothing + soglia": lo smoothing rende le vene più compatte (aumentando l'estensione delle zone ad alta intensità) e la soglia estrae una maschera binaria finale più piena e meno frastagliata.

La binarizzazione conclusiva ($binarizza(volumeGauss, 'manuale', soglia)$) utilizza una soglia che può essere inserita dall'utente; se invece $soglia = -1$, la funzione attiva una stima automatica. In questo caso viene costruita una curva diagnostica: per ciascuna soglia intera da 1 a 250, il volume gaussiano viene binarizzato e si calcolano le componenti connesse; di ciascuna componente si stima il volume (numero di voxel) e si conteggiano solo quelle "grandi" (volume ≥ 1000 voxel). Il risultato è un vettore vecNumCC che descrive quante componenti connesse strutturalmente rilevanti sopravvivono al variare della soglia. Questa analisi è eseguita in parallelo ($parfor$) per contenere i tempi computazionali.

La scelta finale della soglia viene demandata a $calcolaSogliaBinGauss$, che trasforma il problema in una ricerca di un punto caratteristico sulla curva "numero di componenti grandi vs soglia". Poiché tale curva può essere rumorosa, viene prima smussata con regressione locale ($smooth(\ldots,'loess')$) usando un parametro di smoothing (nella chiamata è pari a 100). Sul vettore smussato si calcola poi la derivata discreta tramite differenza finita ($dy = diff(vecProcessed)$), e si individuano i massimi locali di dy con $findpeaks$. La funzione seleziona il primo massimo locale della derivata come punto di transizione iniziale in cui la curva cambia più rapidamente, interpretandolo come una soglia "informativa" che separa un regime di eccessiva frammentazione da uno in cui le strutture iniziano a consolidarsi o a scomparire in modo più sistematico. L'indice del picco viene infine convertito in soglia reale moltiplicando per un fattore di scala step (nel caso in uso pari a 1). Quando richiesto, calcolaSogliaBinGauss produce anche grafici diagnostici della curva originale/smussata e della derivata, utili per validare visivamente la correttezza della soglia stimata.

In sintesi, l'ispessimento realizza una post-elaborazione volumetrica mirata: l'erosione iniziale riduce contributi instabili e rumore sottile, il filtraggio gaussiano ricompone e regolarizza le strutture, e la soglia finale estrae un pattern venoso più compatto e utilizzabile. L'eventuale stima automatica della soglia rende la procedura adattiva rispetto alle variazioni di qualità del dato e di contrasto tra acquisizioni, fornendo un criterio riproducibile basato sulla dinamica delle componenti connesse di dimensione significativa.


*Figura 16: Rappresentazione grafica della matrice volSpesso dopo l'ispessimento.*

## 2.7. Filtraggio delle componenti connesse

Lo scopo di questa fase è quello di eliminare le componenti connesse che non sono delle vene e mantenere invece le vene. Per realizzare questo scopo si è addestrato un modello di Machine Learning, in particolare un classificatore binario per classificare le componenti connesse.

### 2.7.1. Addestramento di un classificatore binario

Per addestrare un modello di classificazione è necessario realizzare un dataset e per realizzare un dataset sono a sua volta necessarie delle features (caratteristiche) da cui il modello deve imparare per fare le previsioni in base al valore delle etichette a loro assegnate.

Per insegnare al modello a riconoscere le vene da altre strutture sono state create delle features basate sulla geometria delle componenti connesse. Le features sono state calcolate basandosi sulle componenti connesse che si trovano nel volume risultato dalla fase di inspessimento. Le features realizzate tengono conto di: le tre dimensioni lungo gli assi principali, le coordinate dei centroidi, la superficie; ci sono poi altre features messe a disposizione da $regionprops3$.

Le features utilizzate si possono raggruppare in due classi:

1) **Feature messe a disposizione da regionprops3 e sono:**
    a. **Volume:** Volume della componente in voxel;
    b. **PrincipalAxisLength:** Lunghezze dei tre assi principali;
    c. **EquivDiameter:** Diametro di un cerchio con la stessa area della cc;
    d. **Extent:** Rapporto tra il volume della componente e il volume del suo bounding box;
    e. **Solidity:** Rapporto tra il volume della componente e il volume del suo convesso;
    f. **SurfaceArea:** Superficie totale della componente;
    g. **Centroid:** Coordinate del centro di massa della componente.
2) **Features derivate combinando o elaborando le feature fornite da regionprops3 e sono:**
    a. **Rapporti tra le dimensioni principali (rapportoAssiYZ, rapportoAssiXZ, rapportoAssiXY):** Valutano le dimensioni delle componenti connesse lungo i tre assi principali;
    b. **Rapporti tra volume e dimensioni principali (rapportoVolumeY, rapportoVolumeX, rapportoVolumeZ):** Indicano come il volume si distribuisce lungo ciascuna dimensione principale;
    c. **Rapporti tra superficie e dimensioni principali (rapportoSurfaceYDim, rapportoSurfaceXDim,**

**rapportoSurfaceZDim):** Indicano la densità superficiale rispetto ai diversi assi;

d. **Rapporto superficie Volume (rapportoSurfaceVolume):** Misura utile per distinguere forme compatte da forme più allungate o frammentate;
e. **Rapporto tra centroide e dimensioni (rappDimXCentX, rappDimYCentY, rappDimZCentZ):** Analizzano la posizione relativa del centroide rispetto alle dimensioni principali;
f. **Distanza Centroide-Pelle (DistCentroidPelle)**;
g. **Rapporto Fine Binarizzazione e Distanza Centroide-Pelle (RappFineBinDistCP)**.

La generazione dei campioni etichettati è orchestrata dallo script ***etichettaCC.m***, che esegue l'intera pipeline di preparazione del volume "da classificare" e la successiva fase di labeling. A partire dal file mat dell'acquisizione, il volume viene sottoposto ai passi già descritti nei paragrafi precedenti (cropping, estrazione del palmo e maschere di supporto, binarizzazione, isolamento del pattern venoso e ispessimento). Il risultato è un volume binario in cui le vene sono rappresentate come strutture attive su sfondo nullo. Prima dell'etichettatura, per ridurre drasticamente il carico di lavoro manuale e limitare l'analisi alle sole strutture più significative, viene applicato un pre-filtraggio dimensionale tramite la funzione ***filtraCC.m***: la funzione estrae le componenti connesse con $bwconncomp$, ne stima la numerosità in voxel mediante $regionprops3(\ldots,'Volume')$ e ricostruisce una maschera contenente esclusivamente le componenti con volume maggiore o uguale a una soglia (nel caso d'uso dello script: 5000 voxel). Questo passaggio elimina in modo sistematico le micro-componenti che, in un contesto venoso, hanno alta probabilità di essere rumore o frammenti non robusti.

La fase centrale di costruzione del dataset è implementata da ***iteraCC.m***, che opera sul volume filtrato e trasforma ciascuna componente connessa in un campione supervisionato. La funzione calcola in primo luogo un insieme di descrittori geometrici 3D tramite $regionprops3$, includendo le proprietà descritte sopra.

L'associazione tra campione e classe viene ottenuta tramite etichettatura interattiva: per ogni componente connessa, ***visualizzaCC.m*** mostra una proiezione sul piano XY sia del volume complessivo sia della sola componente corrente, rendendo l'operatore in grado di valutare rapidamente forma e contesto della struttura. L'utente assegna quindi una label discreta secondo la convenzione implementata: 1 per "vena", 0 per "rumore", 2 per "ignora" (componente esclusa dal dataset) e -1 per annullare l'intera sessione corrente senza salvare risultati. Le componenti marcate come "ignora" vengono rimosse a posteriori dalla tabella finale, mentre in assenza di annullamento

la tabella completa (feature + label) viene salvata in formato CSV per la singola acquisizione. Questo meccanismo produce un dataset strutturato in cui ogni riga corrisponde a una componente connessa e ogni colonna a una feature numerica o all'etichetta di classe, costruendo di fatto la ground truth necessaria per l'apprendimento supervisionato.

Per automatizzare la creazione del dataset su più acquisizioni di uno stesso soggetto, lo script ***creaDatasetUtente.m*** scansiona la cartella dei file mat associati all'utente, seleziona le acquisizioni desiderate tramite una whitelist e invoca $etichettaCC$ per ciascuna di esse, generando progressivamente i CSV per-acquisizione. Successivamente, ***creaDataset.m*** esegue la fusione dei file CSV presenti nella cartella $./dataset/$, concatenandoli verticalmente in un unico dataset globale. Tale unificazione consente di costruire un insieme di addestramento multi-utente/multi-acquisizione più vario e generalizzabile; lo script include inoltre (in forma opzionale) la possibilità di rimuovere specifiche colonne prima dell'addestramento, così da controllare la dimensionalità delle feature e testare configurazioni diverse (feature selection manuale).

L'addestramento del classificatore vero e proprio è realizzato da ***addestraModello.m***. Il dataset (in particolare la versione eventualmente filtrata per colonne, $datasetUnitoColonneFilt.csv$) viene caricato come tabella, rimescolato casualmente per ridurre bias dovuti all'ordinamento dei campioni e suddiviso in training e test set con proporzione 80/20. Coerentemente con l'organizzazione dei CSV prodotti dalla fase di etichettatura, tutte le colonne tranne l'ultima vengono utilizzate come feature numeriche (X), mentre l'ultima colonna è interpretata come label binaria (y). Il modello adottato è un ensemble in bagging di alberi decisionali ($fitcensemble$ con $Method = 'Bag'$), con ottimizzazione automatica degli iperparametri ($OptimizeHyperparameters = 'auto'$), così da ricercare configurazioni più efficaci senza imporre a priori numero di alberi, profondità o altri parametri critici. La qualità del classificatore viene infine stimata sul test set mediante matrice di confusione e accuratezza complessiva; il modello addestrato viene salvato su disco (***modelloRF.mat***) per l'impiego nelle fasi successive della pipeline, dove potrà essere utilizzato per filtrare automaticamente le componenti connesse e isolare, in modo riproducibile, le strutture venose più plausibili.

### 2.7.2. Filtraggio ricorsivo con il classificatore binario

Una volta addestrato il modello è stato utilizzato per fare le previsioni e quindi eliminare tutto il rumore inutile ai fine della creazione del template. La funzione ***filtraComponentiConnesse.m*** effettua come primo passaggio operativo la separazione delle componenti connesse del volume in due insiemi sulla base della soglia $minSize$ (nel caso della chiamata, 5000 voxel). Questo split viene effettuato tramite una

funzione ausiliaria ***dividiCCvolume.m***, concettualmente interpretabile come: calcolo delle componenti connesse, stima del volume di ciascuna CC e ricostruzione di due volumi binari, uno contenente le CC "piccole" ($volume < minSize$) e uno contenente le CC "grandi" ($volume \geq minSize$). Tale divisione riflette un'euristica precisa: le strutture venose di interesse, dopo l'ispessimento, tendono a generare componenti volumetricamente rilevanti, mentre molti elementi di rumore rimangono frammentari e di dimensioni inferiori.

Nel secondo passaggio viene applicata la classificazione supervisionata alle sole componenti grandi. In $filtraComponentiConnesse$ questa operazione è demandata alla funzione ***filtraCC.m***, chiamata utilizzando 3 come parametro di limite/contatore di ricorsione (in pratica un "budget" massimo di tentativi di recupero su strutture ambigue). La funzione $filtraCC$ realizza il filtraggio supervisionato a livello di componente connessa. A partire dal volume binario in ingresso, calcola le componenti connesse 3D ($bwconncomp$) e per ciascuna CC estrae un set di descrittori geometrici tramite $regionprops3$(come descritto in precedenza). A questi attributi base affianca feature derivate. Il vettore di feature così costruito viene quindi fornito al classificatore addestrato (model) per predire se la CC sia compatibile con una vena oppure con rumore; il volume di output viene ricostruito mantenendo solo le componenti classificate come "vena". Per ridurre falsi negativi su strutture complesse, $filtraCC$ integra un recupero ricorsivo controllato: se una componente molto grande viene classificata come rumore, la funzione può invocare recuperaVena (decrementando il contatore $stopRicorsione$) per tentare di separare/ricostruire la componente e recuperare eventuali porzioni venose interne; le parti recuperate vengono sommate al volume filtrato finale. L'output è quindi un volume filtrato (tipicamente riportato in formato uint8 con voxel attivi a 255) che conserva le sole strutture ritenute venose dal modello, includendo ove possibile contributi recuperati da componenti ambigue. A valle della classificazione, le componenti grandi accettate vengono mantenute nel volume filtrato $volumeFCCR$, al quale vengono poi riaggiunte le CC piccole ($volumeFCCR = volumeFCCR \mid volumeCCPiccole$). Questa scelta è intenzionale: anche se l'informazione discriminante principale viene estratta dalle CC grandi, alcune strutture minute possono rappresentare dettagli utili (ramificazioni sottili, porzioni terminali) che non conviene scartare prematuramente.

Il terzo passaggio mira a correggere un limite tipico della binarizzazione volumetrica: la frammentazione del pattern venoso in segmenti discontinui. Tale ricostruzione è realizzata da ***connettiVene.m***, che applica una pipeline specifica basata su smoothing gaussiano e chiusura morfologica. Operativamente, $connettiVene$ converte il volume binario in $uint8$ con valori $(0, 255)$, quindi applica un filtro gaussiano 3D ($imgaussfilt3$, $sigma = 1$) per "diffondere" localmente l'intensità e favorire la fusione di strutture molto vicine. Su questo volume smussato, la funzione stima

automaticamente una soglia di binarizzazione esplorando un range di soglie (1–150 con passo 2) e, per ciascuna soglia, conteggiando quante componenti connesse risultanti hanno volume ≥ 1000 voxel. Il vettore così ottenuto (numero di CC significative vs soglia) viene ulteriormente regolarizzato con smoothing LOESS, e la soglia finale viene scelta in corrispondenza del primo minimo locale della derivata discreta del vettore smussato (interpretato come un punto di stabilizzazione/cambio di regime nella frammentazione). Infine, il volume binarizzato viene sottoposto a chiusura morfologica 3D ($imclose$ con sfera di raggio 3), che colma piccoli gap e "salda" ulteriormente segmenti adiacenti. Il risultato è un volume $volumeConnesso$ in cui la connettività delle vene è tipicamente più coerente rispetto a $volumeFCCR$.

A questo punto, $filtraComponentiConnesse$ introduce un quarto passaggio che separa esplicitamente un insieme di componenti "minuscole" (chiamate palline) dal resto del volume connesso. Il volume viene nuovamente diviso con soglia fissa 1000 voxel. Le palline ($CC \leq 1000\ voxel$) vengono accantonate e preservate, mentre sulle componenti rimanenti si calcola una soglia adattiva di separazione tra "piccole" e "grandi" basata sulla distribuzione dei volumi delle CC: si stimano media e deviazione standard dei volumi (tramite una tabella ottenuta con $regionprops3(\ldots,'Volume')$) e si imposta $soglia = round(std - mean)$; se tale valore risulta non positivo, viene usata come fallback $round(std)$. Questa euristica introduce un criterio di scala che dipende dalla "taglia tipica" delle componenti presenti in quel volume: in acquisizioni dove le vene risultano più continue e grandi, la soglia si sposta; in acquisizioni più rumorose o frammentate, la soglia tende a ridursi. Con tale soglia adattiva, il volume viene diviso ancora una volta in un insieme di CC relativamente piccole (ma non minuscole) e un insieme di CC grandi.

Il quinto passaggio applica nuovamente la classificazione supervisionata, ma questa volta alle CC piccole emerse dopo la fase precedente. Qui il parametro finale è posto a 0, indicando che per le CC piccole si evita l'attivazione della ricorsione: su componenti di dimensioni ridotte la strategia di "recupero" tende ad essere meno informativa e più incline a far proliferare falsi positivi. Il volume viene quindi ricomposto mantenendo tutte le CC grandi (assunte più affidabili dopo connessione) e aggiungendo le CC piccole che il classificatore reputa compatibili con vena.

Nel sesto passaggio viene eseguita una pulizia finale per garantire coerenza dimensionale: il volume ricomposto viene filtrato mantenendo solo le componenti sopra una soglia fissa di 5000 voxel, e il risultato viene convertito in una rappresentazione compatibile con la visualizzazione (impostando i voxel attivi a 255).

Nel settimo passaggio, le palline accantonate in precedenza vengono riunite al volume finale ($volFilt = volFilt \mid volumeCCPalline$), così da non perdere completamente quella porzione di informazione. In altri termini, l'output $volFilt$ è volutamente

sbilanciato verso due estremi: (i) grandi strutture venose affidabili e (ii) un insieme di piccolissime componenti preservate per cautela, mentre le componenti intermedie vengono mantenute solo se hanno contribuito alla formazione di strutture connesse più grandi o se superano i filtri supervisionati.

Il comportamento realmente "ricorsivo" del filtraggio emerge quando si considera la funzione ***recuperaVena.m*** chiamata da $filtraCC$, progettata per gestire un caso critico: una componente molto grande può essere classificata come rumore dal modello perché ingloba contemporaneamente vene e strutture spurie (o perché presenta geometrie atipiche). In questo scenario, invece di scartare l'intera componente, si tenta un recupero guidato. La funzione $recuperaVena$ estrae la singola CC dal volume (mediante $CC.PixelIdxList\{i\}$), applica un inspessimento/riconnessione adattivo tramite ***inspessimentoRecuperoAdatt.m*** e poi divide il risultato in CC piccole e grandi con soglia 1000 voxel. Se dopo questa trasformazione la parte "grande" si separa in più di una componente significativa, ciò è interpretato come evidenza che la CC originale era un aggregato: a quel punto viene rilanciata la classificazione $filtraCC$ sulle sole CC grandi risultanti, propagando un contatore $stopRicorsione$ per evitare cicli e ramificazioni eccessive. Le CC piccole vengono infine riunite al risultato per non perdere dettagli.

La funzione $inspessimentoRecuperoAdatt$ è specificamente tarata per questa fase: esegue un'erosione leggera (sfera r=1) per spezzare ponti sottili e rendere più separabili le porzioni adese, applica smoothing gaussiano 3D e stima una soglia di binarizzazione esplorando 1–150 con passo 3, scegliendo la soglia in corrispondenza di picchi della derivata del numero di CC significative (euristica che cerca un "cambio di regime" utile a separare sottostrutture).

In sintesi, la ricorsione è un meccanismo di rifattorizzazione strutturale di componenti ambigue, che tenta di trasformarle in un insieme di sottocomponenti più "leggibili" dal classificatore.

Complessivamente, $filtraComponentiConnesse$ implementa quindi un filtraggio supervisionato a più livelli che combina euristiche geometriche (soglie dimensionali fisse e adattive), ricostruzione morfologica della continuità (connessione tramite gaussiano + closing) e un recupero ricorsivo mirato (tramite $recuperaVena$ e $inspessimentoRecuperoAdatt$) per ridurre i falsi negativi su componenti grandi complesse. Il risultato finale volFilt rappresenta un compromesso tra selettività (rimozione sistematica del rumore) e robustezza (tentativi controllati di recupero quando la decisione "rumore" potrebbe mascherare porzioni venose reali), producendo un volume più stabile e coerente per la generazione del template e le fasi successive di matching.


*Figura 17: Rappresentazione grafica della matrice volFilt dopo il filtraggio finale.*

## 2.8. Affinamento delle vene

La fase di affinamento ha l'obiettivo di rifinire il volume binario ottenuto dagli step precedenti, riducendo i disturbi residui e migliorando la continuità topologica delle strutture venose, senza compromettere eccessivamente la sensibilità (cioè evitando di eliminare tratti venosi reali). Il flusso è implementato in ***affinaVene.m*** e combina tre strategie complementari: (i) rimozione locale di artefatti con analisi slice-by-slice, (ii) filtraggio globale guidato da feature geometriche e di posizione (in particolare rispetto al palmo), (iii) recupero controllato di porzioni scartate e "filling" selettivo delle zone troppo sottili.

Dopo il caricamento del modello di classificazione (Random Forest, da *modelloRF.mat*), il volume in input viene separato in componenti connesse (CC) "piccole" e "grandi" tramite una soglia volumetrica iniziale (1000 voxel). Le CC piccole non vengono immediatamente eliminate: vengono conservate perché spesso rappresentano frammenti deboli o discontinuità che possono risultare utili nel recupero o nella ricostruzione finale. L'affinamento vero e proprio è invece applicato alle sole CC grandi. Per poter applicare le successive operazioni di affinamento su una sola struttura alla volta, la funzione ricostruisce la componente i-esima in forma di maschera volumetrica logica della stessa dimensione del volume originale. Operativamente si crea un volume booleano inizializzato a false e si imposta a *true* esclusivamente la posizione dei voxel appartenenti alla componente corrente. Poi invoca ***affinaDisturbi.m***, una routine che opera lungo l'asse Y piano per piano (piani

XZ). Su ogni piano non vuoto, $affinaDisturbi$ applica prima un riempimento dei buchi ($imfill$) per rendere le regioni più compatte, poi raggruppa le componenti 2D "vicine" costruendo un grafo di adiacenza basato sulla distanza fra centroidi: due regioni vengono considerate vicine se la loro distanza è inferiore a una soglia proporzionale al diametro equivalente massimo del piano. All'interno di ciascun gruppo, viene quindi separato il rumore "attaccato" alle vene distinguendo, in modo relativo, le aree piccole (inferiori al 10% dell'area massima del gruppo, interpretate come "pallini") dalle aree principali (interpretate come porzione venosa). Infine, sulle CC residue si applica una pulizia morfologica adattiva: un'operazione di apertura ($imopen$) con raggio derivato dall'$EquivDiameter$ della componente (quindi scalato sulla sua dimensione) rimuove appendici sottili e irregolarità; se l'apertura risultasse troppo aggressiva (annullando la regione), la componente originale viene ripristinata. Un'ulteriore selezione post-apertura mantiene solo frammenti con area significativa rispetto al massimo (≥ 50%), così da eliminare residui staccati. L'output di $affinaDisturbi$ è duplice: un volume "ripulito" e un volume di "pallini" separati, che $affinaVene$ conserva per possibili reintegri successivi.

Una volta ottenuta la componente ripulita, $affinaVene$ esegue un'ulteriore divisione in sotto-componenti "vene" e "pezzi" usando una soglia più alta (5000 voxel): l'idea è trattare separatamente ciò che appare strutturalmente robusto da ciò che è più frammentario o dubbio. Sulle "vene" viene poi applicato un filtro supervisionato ($filtraCC$) che usa feature contestuali per mantenere solo le CC classificate come venose dal modello. Le regioni che il modello scarta non vengono però ignorate: vengono accumulate come "pezzi" candidati al recupero, perché potrebbero contenere tratti venosi reali eliminati per eccesso di severità. Prima di accumulare definitivamente il risultato, la componente classificata come vena viene regolarizzata con un filtraggio gaussiano volumetrico ($filtGaussVol$) e con una chiusura morfologica 3D ($imclose$ con sfera di raggio 5), così da colmare micro-interruzioni e aumentare la continuità locale. Iterando su tutte le CC grandi, si costruiscono quindi due volumi aggregati: uno contenente le vene "accettate" ($volumeAffinato$) e uno contenente porzioni dubbie o frammentarie ($volumePezzi$), mentre i "pallini" ritornati da $affinaDisturbi$ vengono reinseriti tra le CC piccole.

Completata la fase locale sulle CC, $affinaVene$ applica una rimozione di rumore a scala globale tramite ***eliminaRumore.m***. Questa funzione non usa un classificatore supervisionato, ma un insieme di feature geometriche e di posizione calcolate per ciascuna CC 3D sufficientemente grande ($Volume \geq minSize$, tipicamente 5000 voxel) e valutate con regole euristiche. In particolare, la componente viene analizzata su piani XZ campionati lungo Y (passo 2): su ogni piano, quando presente, viene individuata la CC 2D e se ne seleziona il centroide più "profondo" (massima coordinata Z nel piano), stimando così una quota rappresentativa della vena su quella

slice. Confrontando tale quota con la quota del palmo senza pelle in (y, x) ($indiciPalmoNoPelle$) si ottiene una distanza palmo–vena, che viene memorizzata lungo Y e sintetizzata tramite statistiche (media, mediana, deviazione standard, percentili, range (differenza tra max e min)). Oltre a ciò, viene stimata una "pendenza media" (variazione lungo Y) dopo smoothing LOESS e derivata discreta, e vengono conteggiati: il numero di piani XZ in cui la componente è presente. Le regole di scarto combinano questi indicatori: vengono eliminate componenti con estensione lungo Y insufficiente (meno di 30 piani), con eccessivo contatto col bordo (oltre il 20% dei piani), oppure con distanza media palmo–vena anomala (oltre μ + σ rispetto alle componenti osservate) associata a bassa pendenza (inferiore a 5), configurazione compatibile con disturbi stazionari e profondi.

Per mitigare la "possibile perdita" dovuta ai filtri, la pipeline introduce un recupero esplicito delle porzioni accantonate. Il volume dei "pezzi" viene ulteriormente ripartito e passato a ***recuperaPezzi.m***, che valuta ogni candidato verificando se la sua reintegrazione migliora la connettività del grafo venoso. $recuperaPezzi$ riceve (i) il volume binario delle vene già accettate ($volume$), (ii) un volume di frammenti scartati/candidati al reinserimento ($componenti$) e (iii) il volume delle componenti connesse piccole ($volumeCCpiccole$, i cosiddetti "pallini"). L'obiettivo non è recuperare indiscriminatamente porzioni eliminate, ma selezionare solo quei "pezzi" che, se reinseriti, aumentano la continuità del pattern venoso, misurando tale miglioramento come riduzione del numero di componenti connesse del volume finale. Prima della valutazione pezzo-per-pezzo, la funzione applica due filtri preliminari per stabilizzare l'insieme dei candidati: se $componenti$ contiene più di 15 componenti connesse, viene applicato un filtraggio supervisionato (tramite $filtraCC$) per ridurre la numerosità dei candidati; successivamente si eliminano i frammenti troppo piccoli mantenendo in $componenti$ solo CC con volume ≥ 1000 voxel e, nei "pallini", solo CC con volume ≥ 100 voxel. In questo modo il recupero lavora su strutture dimensionalmente significative e riduce la probabilità che micro-rumori producano fusioni spurie. La selezione finale avviene iterando su ciascuna CC candidata in componenti. Ogni pezzo viene ricostruito come maschera 3D ($comp$) e descritto tramite centroide e lunghezza dell'asse principale ($PrincipalAxisLength$). Tale grandezza viene usata come scala spaziale per selezionare un sottoinsieme di "pallini" potenzialmente utili come ponti locali: per ogni CC in $volumeCCpiccole$ si calcola il centroide e si include nel set pallini solo se la distanza euclidea dal centroide del pezzo è $\leq PrincipalAxisLength(1)$ del pezzo stesso. A questo punto la funzione valuta l'impatto topologico dell'inserimento del pezzo (insieme ai soli pallini vicini) confrontando il numero di componenti connesse prima e dopo l'unione, dopo un passaggio di stabilizzazione con $filtGaussVol$ applicato sia ai volumi baseline sia a quello unito. In particolare si calcola: $v$, numero di CC del volume vene filtrato; $p$,

numero di CC dei pallini filtrati; e si definisce un totale "atteso" senza fusioni come $t = v + 1 + p\,(vene + pezzo + pallini)$. Si costruisce quindi $volumeUnito = volume\ OR\ comp\ OR\ pallini$ e se ne conta il numero di CC ($cc$) dopo lo stesso filtraggio gaussiano. Il pezzo viene considerato candidato al recupero solo se $t - cc > 1$, cioè se l'unione produce una riduzione di CC superiore a una singola fusione (euristica più selettiva che riduce casi accidentali). Poiché parte della riduzione potrebbe essere dovuta esclusivamente ai pallini (che possono fondersi tra loro o con le vene senza che il pezzo sia realmente determinante), il codice introduce una seconda verifica: vengono conteggiate quante CC finali del $volumeUnito$ corrispondono ancora a pallini selezionati, confrontando i centroidi con una tolleranza del 5% per coordinata ($abs(diff) < 0.05 * coord$). Questo numero ($pr$) rappresenta quante componenti finali siano attribuibili ai pallini; di conseguenza, la condizione conclusiva richiede che anche "al netto" dei pallini rimanga un guadagno di connettività: il pezzo viene recuperato solo se $(v + 1) - (cc - pr) > 1$. Solo in questo caso $comp$ viene aggiunto a volumePezzi. In sintesi, $recuperaPezzi$ reinserisce un frammento non perché "somiglia" a una vena, ma perché la sua aggiunta produce un miglioramento topologico robusto, verificato in due stadi: prima sul volume complessivo (vene+pezzo+pallini vicini), poi escludendo esplicitamente le fusioni imputabili ai soli pallini.


*Figura 18: Rappresentazione grafica della matrice volumeAffinato dopo un primo affinamento.*

Dopo il recupero, $affinaVene$ applica un post-processing finale: smoothing e chiusura morfologica (ancora con sfera di raggio 5) per consolidare la continuità globale, seguiti da una rimozione di CC "corte" tramite ***eliminaCCcorte.m***. In questa funzione, ciascuna componente è descritta da $PrincipalAxisLength$ e $Volume$: vengono mantenute solo quelle con estensione principale lungo Y superiore a 150 e volume

superiore a 10000 voxel, eliminando frammenti troppo brevi per essere plausibili vene robuste nel contesto considerato. Il volume filtrato viene codificato in $uint16$ con voxel attivi a 255, coerentemente con la convenzione di visualizzazione/archiviazione della pipeline, e viene infine sottoposto a un ultimo passaggio di filtraggio tramite modello ($filtraCC$) per una pulizia ulteriore.


*Figura 19: Rappresentazione grafica della matrice volumeAffFilt dopo il filtraggio globale.*

L'ultimo step, mira a migliorare la resa delle vene troppo sottili o con tratti discontinui senza ispessire indiscriminatamente l'intera struttura. Per ogni CC finale, viene eseguito un affinamento locale (richiamando nuovamente $affinaDisturbi$ e scartando eventuali componenti sotto soglia) e si stimano indicatori di spessore tramite ***calcolaFeaturesFilling.m***: la funzione calcola, per ciascuna slice Y non vuota, la media dei diametri equivalenti ($EquivDiameter$) delle CC 2D nel piano XZ, ottenendo un vettore di diametri rappresentativo lungo Y. Da questo vettore si ricava una soglia di "tratto sottile" come media tra mediana e minimo (euristica) e si definisce un elemento strutturante sferico con raggio circa pari a metà dello spessore medio. La componente viene poi scheletrizzata ($bwskel$, con $MinBranchLength = 30$) per ottenere l'asse centrale e si calcola la distanza interna dal bordo ($bwdist(\sim componente)$). I voxel dello skeleton che cadono in zone con distanza inferiore alla soglia vengono identificati come porzioni sottili; solo queste vengono dilatate con l'elemento strutturante e riunite alla componente originale. In questo modo il riempimento agisce localmente dove la vena è più fragile o sottile, riducendo interruzioni e migliorando la continuità del template senza introdurre un "gonfiaggio" generalizzato. Il risultato complessivo ($volumeFilled$) viene infine salvato (se non vuoto) e costituisce l'output dello step di affinamento.


*Figura 20: Rappresentazione grafica della matrice volumeFilled dopo il filling finale.*

# 3. Analisi critica del metodo CCM

Il metodo CCM si è verificato essere piuttosto efficace nell'estrazione del pattern venoso per quegli individui in cui esso è sufficientemente spesso e la vena principale è anche sufficientemente profonda. D'altronde soffre di alcune limitazioni legate soprattutto alle assunzioni fatte a monte dello sviluppo del codice, unite anche a scelte implementative discutibili o poco generali. Lo scopo di questo capitolo è quello di analizzare il metodo in maniera critica, evidenziando le sue debolezze sia dal punto di vista teorico che implementativo, e tentando di proporre delle soluzioni o mitigazioni a tali debolezze.

## 3.1. Analisi del passo relativo all'estrapolazione del volume contenente vene

Nella funzione $estrapolaVolumeVene$ nella costruzione del volume di quote chiamato $SURF$, viene impiegata la soglia $tresh$. Quest'ultima è utilizzata per creare una maschera binaria a partire da $M$ . Quindi dove $M$ è sotto soglia $SURF$ viene azzerato, lasciando indici di profondità solo nei voxel "validi". Il problema è che questa soglia, come diverse altre soglie del metodo, viene fissata a un valore specifico, in questo caso 64. Il problema insito nel fissare a monte una soglia a un valore specifico risiede nel fatto che andrebbe dimostrata l'efficacia di tale valore in diverse situazioni e casistiche. Oppure tale soglia dovrebbe essere adattiva, ovvero dovrebbe essere calcolata in base a caratteristiche specifiche del volume sotto analisi e quindi dovrebbe variare al variare del volume. Questa seconda possibilità sebbene sia più dispendiosa in termini di costo di calcolo, è sicuramente la scelta più robusta e generale. Lo standard in letteratura per segmentare mano/palmo in immagini venose prevede l'utilizzo di soglie globali tipo $Otsu$, proprio perché automatiche, semplici e robuste. Inoltre tali tipologie di soglie in $MATLAB$ sono facilmente utilizzabili poiché esiste una funzione predefinita che le implementa ovvero $graythresh$. Discorso simile potrebbe essere fatto per le altre due variabili $offsetPalmo = 200$ e $offsetPelle = 10$ le quali vengono passate da $estrapolaVolumeVene$ alla funzione ausiliaria $calcolaMaschere$. Nella prima iterazione ($z == 1$), per ogni colonna (y, x) viene stimata una "altezza" del palmo lungo z ($altezzaPalmo$, trovata come ultimo non-zero nella colonna) e si crea: $mascheraNeroPalmo(y, x, 1{:}\, altezzaPalmo - offsetPalmo) = true$. L'effetto concettuale è: tenere solo una banda di spessore fisso (pari a $offsetPalmo$ slice) "vicina" alla porzione di palmo considerata utile, e mascherare il resto. In altre parole, $offsetPalmo$ definisce quanta profondità (in slice) viene considerata pertinente per il pattern venoso, scartando regioni che potrebbero essere solo "pre-palmo o zone non informative o rumore". Il problema principale di questa euristica è che è basata sul numero di slice, se invece fosse convertita in mm si potrebbero utilizzare range

anatomici tipici come range plausibili: la cute del palmo è notoriamente più spessa di altre zone (ordine ~0.8–1.4 mm come grandezza complessiva riportata) (Davidson). In $calcolaMaschere$ viene calcolata, per ogni colonna (y, x), la prima slice in cui $mascheraAcqua$ è true lungo z ($indiciAcqua$). Poi, per $offsetPelle$ iterazioni, l'indice viene "spostato" e si accende progressivamente mascheraAcqua su uno strato aggiuntivo. In pratica questa parte implementa una dilatazione 1D lungo z della maschera acqua superficiale: l'effetto è scartare un certo spessore superficiale interpretato come "strato non informativo", rendendo più conservativa la regione valida. Questa costante potrebbe essere in realtà tolta oppure al massimo potrebbe anch'essa essere utilizzata come per la precedente tramite la rispettiva versione dimensionale in mm.

## 3.2. Analisi del passo relativo alla binarizzazione del volume

La funzione $effettuaBinarizzazione$ chiama moltissime funzioni ausiliarie, ed è in generale piuttosto articolata quindi i punti da analizzare sono diversi.

Suddetta funzione chiama la funzione $calcolaVecFineBin$ la quale esegue diverse operazioni, tra questa vi è la binarizzazione piano per piano del volume. Le binarizzazioni dei singoli piani sono delegate alla funzione $binPianoSingolo$. Questa funzione di fatto filtra il piano con una soglia di binarizzazione (molto alta) cioè seleziona solo i pixel quasi saturi. Questo perché si era in un momento precedente invertito il contrasto del volume, e quindi le vene profonde si trovano ad avere valori piuttosto alti di intensità. Si ha quindi $soglia = 254$ che ancora una volta è una euristica e per gli stessi motivi descritti sopra è difficilmente accettabile; inoltre è eccessivamente restrittiva come soglia. Una possibile soluzione in questo caso potrebbe essere quella di rendere adattiva al piano la soglia tramite una $quantile\ thresholding$. Cioè si costruisce l'istogramma delle intensità dei voxel, si calcola la percentuale cumulata e si sceglie una soglia $T$ tale che la "coda alta" (i voxel con intensità $> T$) siano al massimo il $p\%$. Il valore $p$ potrebbe essere calcolato basandosi su una serie di volumi di test. Sempre nella funzione $calcolaVecFineBin$ una volta che viene popolato il vettore contenente per ogni valore di fine il numero di componenti connesse (chiamato $vecCC$), vengono trovati i vari "picchi" presenti nel vettore e sostanzialmente per la selezione del valore ottimo ci si muove come segue: se si hanno almeno due picchi si usa la media dei primi due; se si ha un solo picco si usa quest'ultimo; se non si hanno picchi si usa $NaN$. Tale meccanismo presuppone implicitamente che: i primi due picchi siano "quelli giusti" (escludendo la possibilità che nelle cc ci sia molto rumore); il punto "buono" stia nel mezzo di questi due picchi; e che i picchi siano comparabili. Nessuna di queste ipotesi è garantita, soprattutto se $vecCC$ ha una alta variabilità (come poi di fatto accade). Invece di determinare il valore "ottimo" di fine a partire dai primi due picchi di $vecCC$, è possibile adottare un criterio

di stabilità topologica, più generale e motivabile: poiché $vecCC(f)$ rappresenta il numero di componenti connesse ottenute al variare del parametro $f = fine$, essa può essere interpretata come una curva dipendente da una filtrazione (aggiunta progressiva di voxel al crescere di f). In questa prospettiva, una scelta robusta del parametro non è un singolo punto arbitrario (picco), ma un intervallo in cui la descrizione topologica cambia poco al variare del parametro, cioè una regione "stabile" rispetto a piccole perturbazioni di f (e, indirettamente, anche rispetto a rumore e variazioni di contrasto). Operativamente si introduce una misura di sensibilità locale, ad esempio $Delta(f) = abs(vecCC(f+1) - vecCC(f))$, e si ricerca un intervallo sufficientemente lungo $[f_a, f_b]$ in cui $Delta(f)$ rimanga sotto una soglia piccola (plateau); a quel punto il valore finale $idOptPiano$ può essere scelto: (i) al centro del plateau, per massimizzare la distanza dai bordi e quindi la robustezza rispetto a piccole variazioni di f, oppure (ii) all'inizio del plateau come scelta conservativa che evita di includere voxel aggiuntivi potenzialmente rumorosi. Questa strategia è coerente con l'idea che le caratteristiche topologiche persistenti su un intervallo di filtrazione siano più affidabili e meno sensibili a perturbazioni.

La funzione $effettuaBinarizzazione$ chiama in prima istanza la funzione $binarizzaVolume$, passandogli il valore 0 sia per $sogliaIniziale$ che per $sogliaFinale$, con questa conformazione dei parametri suddetta funziona lavora in modalità "iniziale". In tale modalità chiama anzitutto la funzione $binManualePiano$, la quale effettua le medesime operazioni della funzione $binPianoSingolo$. E come discusso in precedenza, effettua un filtraggio con una soglia euristica in questo caso imposta al valore 250. Ovviamente anche in questo caso valgono le stesse considerazioni e le stesse proposte di soluzione discusse sopra. Sempre la funzione $binarizzaVolume$ chiama la funzione $binOffsetVena$, la quale ha lo scopo di propagare la binarizzazione lungo z a partire dalla zona delle vene in un singolo piano XZ. L'effetto pratico è: "tenere la binarizzazione solo in una finestra di profondità sotto la prima quota venosa osservata". Per farlo si basa sulla quota(z) del centroide della CC più superficiale e a questa quota aggiunge un offset ($offsetVene = 60$). Questa strategia in generale ha diverse criticità:

- La quota più superficiale non è scelta in maniera ottima poiché: usare il minimo dei centroidi è fragile come soluzione, se esiste anche una sola CC piccola di rumore più superficiale, quella diventerebbe il riferimento e sposterebbe la finestra nel posto sbagliato. In genere qui ci si aspetterebbe un filtro (per area/forma) prima di scegliere il riferimento.
- La quota più profonda non è scelta in maniera ottima poiché: dalla documentazione del sistema di acquisizione sappiamo che lo spessore reale che intercorre tra uno slice e il successivo lungo z è pari a $0.2mm$. Questo unito al fatto che il volume sotto analisi è composto di 320 slice lungo z porta a calcolare

uno spessore totale pari a: $320 slice \times 0.2\, mm/slice = 64\, mm$. Facendo lo stesso calcolo per l'offset scelto in questa funzione si ottiene: $60\, slice \times 0.2\, mm/slice = 12\, mm$. Quindi, questa scelta per quanto riguarda l'offset, che va poi a influenzare la quota più profonda, anche se ragionevole in termini di mm rispetto al volume totale rimane una scelta euristica non giustificata da analisi. In questo caso una CC troppo in profondità potrebbe portare il codice ad andare in errore, poiché farebbe un accesso a un elemento della matrice non esistente, non essendoci alcun controllo che impedisca lo sforamento. Se anche non si presentasse il caso estremo descritto sopra, una CC più profonda della media potrebbe comunque portare il codice a considerare una finestra poco vantaggiosa.

Questa funzione, come visto, anche se pare funzionare per i casi sotto analisi, non garantisce il suo funzionamento per nuovi casi, e in generale non può né scalare né generalizzare. In questo caso più che una semplice correzione sarebbe necessario riscriverla in maniera più robusta. Si potrebbe anzitutto introdurre la verifica sulla dimensione della CC che serve a stabilire la z di partenza. Infine si potrebbe implementare un offset adattivo che continui ad includere slice finché la "presenza venosa" (continuità di CC) resta sopra una soglia, invece di fermarsi sempre a 60. Resterebbe da capire, in maniera data-driven, quale è un volume accettabile per la verifica della CC iniziale e quale è una soglia accettabile per terminare l'inclusione delle slice.

La funzione $effettuaBinarizzazione$ chiama in seguito la funzione $separaStrutture$ la quale effettua tutta una serie di operazioni morfologiche, atte a filtrare e pulire il volume. Al suo interno però chiama la funzione ausiliaria $binarizza$ la quale effettua una binarizzazione del volume con la soglia che gli viene passata. In questo caso la soglia è pari a 100 e nonostante sia decisamente più permissiva e quindi possa causare meno esclusioni rispetto a quelle discusse in precedenza, vale comunque il discorso fatto sulla sua mancanza di generalità. Anche in questo caso si potrebbe applicare una soluzione adattiva simile a quella proposta in precedenza.

La funzione $effettuaBinarizzazione$ chiama anche la funzione $calcolaMinDistVenePalmo$, la quale come si può intuire dal nome calcola la minima distanza che intercorre tra il palmo e una vena, ovvero la distanza dal palmo della vena più superficiale del volume. Per ottenere questo risultato effettua diverse operazioni, tra queste filtra le vene "affidabili" da quelle "non affidabili". Il codice considera affidabili le vene che lungo z hanno più di 3 voxel, infatti imposta la seguente variabile $numVoxelVena = 3$, e la usa per filtrare le vene. Anche quest'ultima è una euristica, e per quanto sia "razionale", non vi è alcuna evidenza che una vena non possa avere uno sviluppo lungo z inferiore a 3 voxel. Tale vena potrebbe infatti essere un capillare, oppure potrebbe essere stata degradata dalle precedenti

operazioni morfologiche. Una possibile soluzione prevede il filtraggio basato sull'area della CC, tale area potrebbe infatti essere convertita in una misura in mm e si potrebbe utilizzare una media dell'area della sezione dei capillari come soglia di filtraggio.

La funzione $effettuaBinarizzazione$ chiama poi la funzione $calcolaSogliaIniziale$ la quale cerca ti calcolare la soglia iniziale di binarizzazione ottima. In tale funzione ci sono due punti critici:

- Sweep delle soglie (250 → 210): non c'è alcuna motivazione data-driven, è un intervallo fissato a mano su un'immagine 8-bit (0…255), quasi certamente frutto di prove empiriche su un certo dataset;
- Il meccanismo per il calcolo della variabile $idDerMinNear$, ovvero: massimo della derivata vicino al massimo globale, poi minimo derivata vicino a quel massimo locale. Non essendo giustificata come strategia è poco chiara, oltre che instabile e non basata su alcuna motivazione analitica valida.

Le possibili soluzioni studiate, per i problemi sopra evidenziati, sono le seguenti:

- Invece di provare delle soglie in un intervallo stabilito con una euristica, paradossalmente, una possibile soluzione potrebbe essere quella di provare tutte le possibili soglie. Questo ovviamente sarebbe fattibile se le operazioni da fare con la suddetta soglia fossero basse in termini costo computazionale. Sfortunatamente in questo caso le operazioni sono piuttosto intensive e quindi bisogna escogitare una ottimizzazione come la seguente: si potrebbe fare uno sweep grossolano (step 5 o 10), trovare dove la metrica cambia regime, per poi rifinire con step 1 solo in quel range.
- Il problema è "trovare un punto di compromesso" sulla curva #CC vs soglia: una soglia troppo alta implica perdere strutture; mentre troppo bassa implica includere rumore/artefatti. Per questi problemi, in letteratura si usano spesso algoritmi di knee detection. In questo caso un algoritmo di knee detection potrebbe essere applicato come segue: si definisce la funzione $m(t) = \#CC_{\geq 2000}(t)$ dove t sono le soglie, ordinate in modo crescente; si normalizzano ambo gli assi in modo che abbiano valori compresi nell'intervallo $[0, 1]$; si cerca il ginocchio con un algoritmo di knee detection; la soglia corrispondente al ginocchio sarà la soglia cercata. La strategia appena descritta oltre a essere standard garantisce di trovare davvero un punto di transizione nei dati analizzati.

Infine la funzione $effettuaBinarizzazione$ chiama la funzione $binarizzaVolume$ passandogli questa volta dei valori diversi da 0 sia per $sogliaIniziale$ che per $sogliaFinale$, con questa conformazione dei parametri suddetta funziona lavora in modalità "finale". In tale modalità, tra le varie operazioni che effettua, chiama la

funzione $calcolaCostanteDiametroVene$. Quest'ultima stima automaticamente il parametro $costanteDiametroVene$ provando diversi valori e scegliendo quello che produce un andamento "stabile" nel numero di componenti connesse significative ($CC$) dopo binarizzazione incrementale. La costante $costanteDiametroVene$ viene stimata come parametro di adattamento della profondità massima impiegata dalla binarizzazione incrementale. In particolare, si considera una famiglia di binarizzazioni ottenute incrementando il parametro $fine$ rispetto a un valore di riferimento $max(vecFine)$ tramite un offset intero $t \in [0, K]$. Per ogni $t$ si valuta il numero di componenti connesse tridimensionali "significative" (con $volume \geq 3000\ voxel$), ottenendo una curva $f(t)$ che descrive l'evoluzione della complessità topologica della segmentazione al variare della profondità considerata. La strategia di selezione di un punto di compromesso da utilizzare è empirica e non standard. Quello che viene fatto è quanto segue: vengono trovati massimi e minimi della funzione smussata; vengono concatenati in un unico vettore; se il numero di diametri validi passati dall'esterno alla funzione sono sufficienti in numero, in questo caso sufficiente vuol dire maggiore di 20 e il vettore precedente ha almeno un elemento si calcola il massimo del vettore precedente; se è rispettata solo la seconda condizione si calcola il minimo del vettore precedente; se infine nessuna condizione è rispettata si calcola il massimo del vettore che contiene il numero delle componenti connesse. Una strategia alternativa, più robusta e soprattutto più standard, parte dal fatto che la selezione del parametro viene formulata come problema di scelta di un punto di compromesso (knee) sulla curva $f(t)$. Si normalizza $f(t)$, si applica uno smoothing robusto (per ridurre oscillazioni dovute a discretizzazione e rumore) e si individua automaticamente il punto di ginocchio tramite un algoritmo dedicato (es. Kneedle), che fornisce il valore $t$ corrispondente al miglior compromesso tra incremento di informazione e saturazione/instabilità del comportamento del sistema.

Infine per quanto riguarda il funzionamento dell'algoritmo nel suo complesso risulta necessario specificare quanto segue: seppur vero che la distanza minima che intercorre tra il palmo e i primi voxel pari a zero corrisponde a una vena questo non permette di individuare delle vene che si trovano al di sotto di questa distanza. Quindi se ci sono più vene profonde a livelli diversi verrà presa in considerazione solo quella più in superfice tagliando le altre.

## 3.3. Analisi del passo relativo all'inspessimento del pattern venoso

Nella funzione *inspessimento*, tra le varie operazioni effettuate, c'è quella del calcolo della soglia ottima. Questo calcolo viene fatto a partire dal vettore contenente per ogni soglia il numero di componenti connesse (CC). A effettuarlo è la funzione

$calcolaSogliaBinGauss$ chiamata dalla funzione precedente. Questa funzione in primis effettua uno smoothing del vettore originale. Successivamente calcola la derivata di quest'ultimo vettore, ne calcola i picchi, e ne prende il primo. Questo primo massimo una volta arrotondato è di fatto la soglia che viene scelta come ottima. Questa è di fatto una euristica, anche se razionale. Si sta cercando un punto di transizione sulla curva $\#CC - soglia$ usando un indicatore semplice (la pendenza). Se si prende il primo massimo locale di $dy$, si sta scegliendo la prima transizione marcata: è di fatto una scelta conservativa nel senso che tende a fermarsi presto, prima di entrare in soglie più aggressive che potrebbero eliminare parti utili (es. vene sottili). Un possibile miglioramento è il seguente: Invece di $maxLocalIdx(1)$(ovvero il primo massimo), si prende il picco con $prominence$ massima ovvero il massimo globale di $dy$. In questo modo si individua la transizione dominante, non la prima increspatura. È una modifica minima che però può introdurre un comportamento nettamente più robusto e sensato nella strategia di selezione della soglia ottima.

## 3.4. Analisi del passo relativo al filtraggio delle componenti connesse

La funzione $filtraComponentiConnesse$ filtra e ricostruisce le componenti connesse (CC) del volume venoso. Per farlo effettua ricorsivamente diverse operazioni morfologiche e predizioni con un modello Random Forest. È in sostanza una funzione piuttosto articolata, attorno alla quale orbitano diverse funzioni ausiliarie.

La seguente catena di chiamata di funzione:

$$filtraComponentiConnesse \rightarrow filtraCC \rightarrow recuperaVena \rightarrow inspessimentoRecuperoAdatt$$

Porta all'esecuzione della funzione $inspessimentoRecuperoAdatt$ la quale soffre di due problemi già ampiamente discussi in questo capitolo, ovvero:

- Euristica sull'intervallo delle soglie da considerare in questo caso: $start = 1, stop = 150, step = 3$;
- Selezione del punto di transizione tramite calcolo della derivata ed estrazione del massimo n-esimo;

Per entrambi questi problemi il suggerimento di miglioramento rimane quello già discusso e perfettamente applicabile anche in questa parte del codice.

Dei soliti due problemi appena ridescritti soffre anche la funzione $connettiVene$ chiamata direttamente da $filtraComponentiConnesse$ in un passo successivo. Anche in questo caso le soluzioni proposte sono le stesse.

Nella funzione $filtraComponentiConnesse$ successivamente viene calcolata una soglia utilizzata per la divisione del volume in due volumi denominati $volumeCCPiccole$ e $volumeCCGrandi$. Questa soglia è per l'n-esima volta calcolata con una euristica non presente in letteratura e inoltre difficilmente giustificabile, ovvero:

$$soglia = round\big(std(t.Volume) - mean(t.Volume)\big).$$

Se la precedente espressione porta a un risultato negativo la soglia viene ricalcolata come segue:

$$soglia = round\big(std(t.Volume)\big).$$

Per come è implementata, spesso equivale semplicemente a usare $std(Volume)$ come soglia, con un "aggiustamento" poco interpretabile nei casi di dispersione estrema. In questo caso tale strategia potrebbe essere sostituita con una strategia basata su quantili ad esempio: considera piccole le CC sotto il 20° percentile dei volumi. Tale percentile potrebbe anche essere calcolato in maniera data-driven per ottenere un risultato migliore ed aderente allo specifico volume.

## 3.5. Analisi del passo relativo all'affinamento del pattern venoso

La funzione $affinaVene$ effettua diverse operazioni atte a rimuovere disturbi residui, recuperare pezzi e aumentare lo spessore delle vene all'interno del volume contente il pattern venoso. Tra queste operazioni chiama anche la funzione $eliminaRumore$, la quale in uno dei suoi passi crea una serie di soglie euristiche le quali vengono poi usate per filtrare il rumore dalle vene. Queste soglie in assenza di alcuna giustificazione teorica sono non standard e quindi in linea di principio sono poco adatte a scalare e generalizzare su volumi che non siano quelli sotto analisi. Le soglie euristiche sotto analisi sono le seguenti:

$$sogliaDist = mu + sigma;$$

$$sogliaPend = 5;$$

$$sogliaPiani = 30;$$

Quest'ultime andrebbero tutte sostituite con soglie più generali e fondate sui dati. Delle possibili soglie più robuste delle precedenti sono:

$$sogliaDist = median(d) + 2.5 * mad(d, 1);$$

$$sogliaPend = median(p) + 2.5 * mad(p, 1);$$

$$sogliaPiani = \max\big(round(0.05 * yDim), prctile(n, 10)\big);$$

Le soglie sono state rese più robuste sostituendo statistiche sensibili agli outlier (media e deviazione standard) con misure robuste. In particolare, per le feature scalari (distanza media palmo–vena e pendenza media) si impiega una soglia del tipo $median + k \cdot MAD$, dove MAD (Median Absolute Deviation) è la mediana degli scarti assoluti dalla mediana e fornisce una stima della dispersione poco influenzata da componenti anomale. In questo modo, valori "eccessivi" rispetto al comportamento tipico vengono identificati in modo stabile anche in presenza di rumore. Per la soglia sull'estensione lungo Y (numero di piani XZ coperti), la costante assoluta viene sostituita con un criterio scalabile e data-driven: $max(0.05 \cdot yDim, P_{10})$, che impone sia una lunghezza minima proporzionale alla dimensione del volume sia l'eliminazione delle componenti più corte rispetto alla distribuzione osservata.

Inoltre nella funzione $affinaVene$ viene imposta una soglia di spessore che viene utilizzata per identificare, lungo lo skeleton della componente venosa, i tratti localmente più sottili (stimati tramite $distance\ transform$ interna) da rinforzare con una dilatazione selettiva. La scelta $sogliaSpessore = (median(vecDiam) + min(vecDiam))/2$ è un'euristica che pone la soglia tra uno spessore tipico (mediana, robusta) e il caso più critico (minimo), con l'obiettivo di ispessire solo la "coda bassa" dei diametri senza gonfiare l'intera vena. La presenza del minimo rende tuttavia la soglia sensibile a valori estremi, per cui alternative più robuste possono usare percentili bassi o statistiche robuste (es. mediana e MAD), così come ampiamente discusso in precedenza. Una possibile soglia che rispetta quanto detto potrebbe essere:

$$sogliaSpessore = median(r) - 1.5 * mad(r, 1);$$

In questi casi una soglia che sia data-driven e resistente agli outlier è sempre da preferire perché: è autoesplicativa; si adatta a più casistiche ovvero è più generale; e infine si comporta correttamente in presenza di valori estremi dei dati.

# Bibliografia

Davidson, M. W. (s.d.). *Palmar Skin*. Tratto da MOLECULAR EXPRESSIONS: https://micro.magnet.fsu.edu/primer/anatomy/brightfieldgallery/palmarskin10xsmall.html