Capitolo 4

Approccio computazionale

In questo capitolo vogliamo descrivere le tecniche usate nel capitolo successivo per la determinazione della dimensione frattale del perimetro dei ghiacciai. Dal momento che i dati a nostra disposizione relativi ai ghiacciai sono contenuti in uno shapefile, file di formato poco conosciuto per chi non si occupa di studi cartografici, vogliamo dare qualche indicazione sulla loro struttura, su quale sia il loro contenuto e su come possiamo estrarre le informazioni da questo tipo di formato.

Le tecniche usate per l'analisi frattale fanno uso dell'elaborazione di immagini. Matlab mette a disposizione un tool di image processing semplice da usare, quindi abbiamo deciso di adottare questo linguaggio per generare l'algoritmo che si occupa dell'analisi frattale.

4.1 Shapefile

Trattiamo brevemente la descrizione del formato shapefile, come importarlo su matlab e come si può aprire con QGIS. Lo shapefile è un formato vettoriale in grado di immagazzinare informazioni geometriche e attributi riguardanti linee, punti, poligoni e oggetti più complessi, ed è composto da un minimo di 3 file:

Main file (.shp) . È il file principale (file binario ad accesso diretto con lunghezza dei record variabile) ed è organizzato in un file header e in record, nei quali sono conservate le informazioni sugli oggetti geometrici. Il primo contiene informazioni tecniche generiche, tra cui il tipo di forma dei record contenuti nel file (che deve essere uguale per tutti gli elementi geometrici che si vogliono descrivere): nel nostro caso è di tipo PolygonZ, cioè descrive forme poligonali chiuse, comprendendo non solo le coordinate dei loro vertici su un piano (x,y),

ma anche la loro altezza z[1]. Di seguito all'header file vengono riportati i record, costituiti ciascun da un record header (informazioni tecniche) accoppiato a un record content, che contiene le informazioni geometriche per il poligono (posizione vertici, numero...).

Index file (.shx) . Anche questo è organizzato in file headers e record: ciascuno di questi contiene l'offset (posizione in byte rispetto all'inizio del file) dei rispettivi record del main file e la loro lunghezza in byte.

dBase file (.dbf) . Contiene attributi generici comuni a tutte le forme, che non sono compresi nel main file.

Un altro file importante che viene accoppiato a questi, ma non indispensabile, è il file di proiezione (.prj), che contiene le specifiche del sistema di riferimento usato per ottenere i vertici nel main file. Comunque, per i nostri scopi, non è necessario conoscere le proprietà tecniche di questi tipi di file: è sufficiente importarli su Matlab, che ne trasforma il contenuto in una struct. Tutti i programmi che costruiscono lo shapefile a partire da questi tre file usano la convenzione che essi devono avere lo stesso nome, differeziandosi quindi per l'estensione. Per una descrizione completa del formato shapefile vedere [5].

4.1.1 QGIS

Come primo passo, abbiamo aperto lo shapefile usando QGIS: questo programma ad interfaccia grafica permette di aggiungere uno shapefile come layer sul progetto e visualizzare le forme descritte (vedi figura 4.1). Inoltre permette di convertire il file in geometrie differenti o estensioni differenti. Ad esempio, abbiamo convertito lo shapefile in un file .kml per visualizzare i ghiacciai su Google earth (vedi figura 4.2) e abbiamo dovuto cambiare la geometria dello shapefile da PolygonZ a Polygon per poterlo importare su Matlab (questa conversione non comporta comunque una perdita di dati, dal momento che abbiamo verificato le coordinate z di ogni punto essere nulle). Ricordiamo, infine, che uno shapefile non contiene informazioni sulla forma delle linee che collegano i vertici nel file: si suppone che le figure vengano costruite collegando vertici con segmenti di retta. Per ottenere figure curvilinee, l'unico modo è considerare una serie di vertici molto vicini tra loro.

4.1.2 Importare shapefile su Matlab

Per importare uno shapefile su Matlab come struct, è sufficiente usare la funzione shaperead('nome-file'), che prende come argomento il nome o il percorso assoluto

[1]Tuttavia, sfruttando il pacchetto di phyton "pyshape", abbiamo potuto scoprire che le coordinate z erano tutte nulle. Per questa verifica è stato necessario python, dal momento che Matlab non supporta la geometria PolygonZ.

Figura 4.1: Immagine di una porzione del dataset di ghiacciai, ottenuta dall'apertura dello shapefile tramite QGIS.

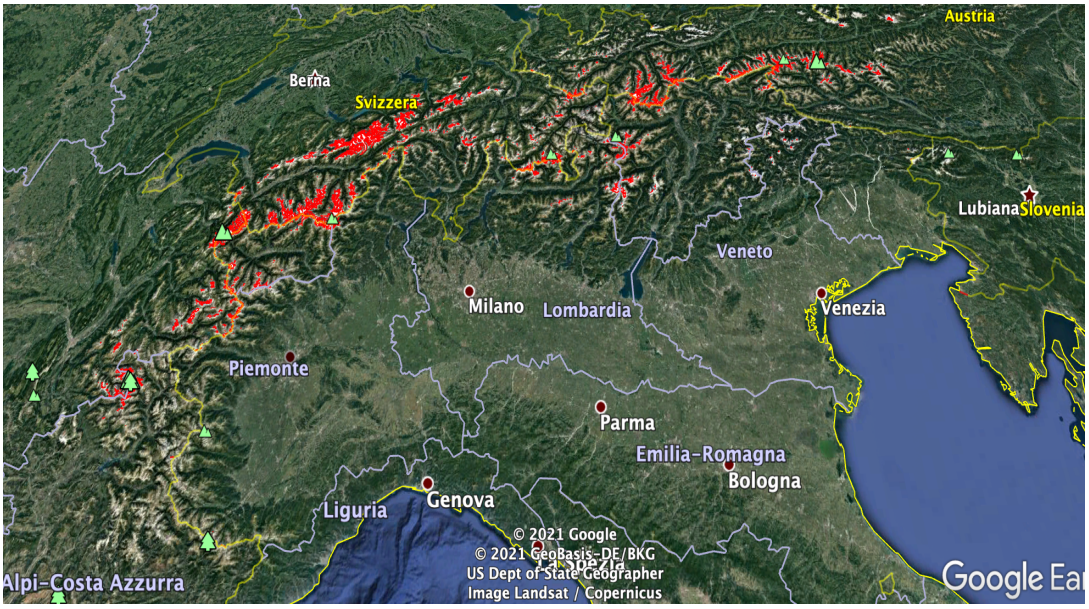


Figura 4.2: Immagine dell'intero dataset di ghiacciai, ottenuta dall'apertura del file .kml tramite Google Earth. I ghiacciai sono i profili in rosso.

o relativo del file .shp: nella medesima cartella in cui è presente questo file, devono anche posizionarsi almeno i file .shx e .dbf, ma è opportuno avere anche il file .prj. Nel nostro caso, shaperead restituisce un array di struct, che assegnamo ad S (S=shaperead('nome-file')), in cui ogni elemento dell'array è uno struct che descrive un ghiacciaio[2]: esso contiene, come campi, tutti gli attributi dello shapefile. I campi di cui faremo uso maggiore sono:

Boundingbox : è una matrice 2×2 in cui la prima riga contiene le coordinate x (prima colonna) e y (seconda colonna) minime tra quelle appartenenti ai vertici del ghiacciaio (definisce il vertice a sud-ovest del box che contiene il ghiacciaio) mentre la seconda riga le coordinate massime (definisce il vertice a nord-est del box che contiene il ghiacciaio).

X : coordinate x dei vertici dei punti che tracciano il perimetro del singolo ghiacciaio.

Y : coordinate y dei vertici del singolo ghiacciaio.

Riportiamo di seguito la lista di tutti i campi dell'array di struct S (una struct per ogni ghiacciaio):
S = 4395×1 struct array with fields:

Geometry
BoundingBox
X
Y
GLACIER_NR
LAT
LON
AREA_KM2
ELEV_MIN
ELEV_MAX
ELEV_MEAN
ELEV_MED
SLOPE_MEAN
ASP_MEAN
SECTOR_NR
SEC_NAMES
Tile_Name
S2_footpri
Date
Funding

[2]Si accede al singolo elemento (ghiacciaio) i-esimo dell'array di struct S con la notazione S(i).

Analyst
O1_REGION
O2_REGION
Country
Footpr_ID

Deduciamo di avere un campione di 4395 ghiacciai. È opportuno anche sottolineare che il campo GLACIER_NR specifica la numerazione del ghiacciaio partendo da zero, mentre l'indice che permette di accedere ai singoli ghiacciai-struct $i \Rightarrow S(i)$ può assumere solo valori positivi, cioè numera i ghiacciai partendo da 1; vale quindi la corrispondenza $S(i) = S(S(i).GLACIER_NR + 1)$. Dalla guida [20] si apprendono altre caratteristiche sugli array di struct restituiti da shaperead. Vengono suddivisi in due categorie: mapstruct se il sistema di riferimento è cartesiano (vertici nello shapefile in coordinate cartesiane); geostruct se il sistema di riferimento nello shapefile usa latitudine e longitudine per indicare la posizione dei vertici. Per capire quale sia il sistema di riferimento nello shapefile, può essere usata la funzione shapeinfo('nome-file'): se il campo "CoordinateReferenceSystem" risultante dall'esecuzione di shapeinfo() è indicato come un oggetto "projcrs", allora siamo in presenza di una mapstruct, mentre se è indicato come "geoscrs", siamo in presenza di una geostuct. Un'ultima importante informazione da conoscere riguardo ai campi di array X e Y è che essi contengono dei valori NaN, utilizzati per separare diverse parti disconesse dei poligoni: ad esempio, se sono presenti dei buchi nel ghiacciaio, un valore NaN[3] di X e Y li separa dal resto dei punti del perimetro esterno. Come è vero anche che ogni punto finale di ogni campo X e Y è un NaN (in questo caso separa due poligoni diversi) e il penultimo punto è uguale al primo (poligono chiuso).

Ancor prima di importare shapefile su Matlab è spesso utile conoscere le sue proprietà, ottenibili tramite la funzione shapeinfo('nome-file'); segnala anche se il formato non è supportato.

4.2 Tecniche per il calcolo della dimensione frattale

In questa sezione discutiamo come sia possibile ottenere la dimensione frattale dei perimetri dei ghiacciai in Matlab. Innanzitutto specifichiamo che andremo a calcolare la dimensione di box-counting, trattata nella sezione 3.2.4: infatti la variante 4 di questa definizione di dimensione si presta bene all'implementazione di algoritmi in grado di calcolarla. Verrà esposto prima il procedimento standard, che sfrutta l'elaborazione delle immagini, poi una tecnica che lavora invece sulle coordinate dei vertici per generare l'immagine binaria (matrice di soli uni e zeri).

[3] "Not a Number", valore che non identifica nessun tipo di dato, restituito solitamente da calcoli errati o impossibili.

4.2.1 Procedimento standard

L'obiettivo è quello di ricavare la dimensione frattale di box-counting D del perimetro dei ghiacciai usando la regressione lineare 3.3. Quindi dobbiamo trovare il modo di suddividere l'immagine di ogni ghiacciaio tramite una griglia di lato δ variabile e, per ogni valore di δ, ottenere il numero $N(\delta)$ di quadrati intersecati dal perimetro, poi inserire i valori $\delta_1, \delta_2, ..., \delta_n \Rightarrow N(\delta_1), N(\delta_2), ..., N(\delta_n)$ nella regressione per ottenere D.

Una possibile soluzione al problema di applicare il metodo box-counting ai perimetri è quella di generare l'immagine di ciascun ghiacciaio i-esimo, salvarla e poi importarla nuovamente su Matlab usando la funzione imread("nome-file-immagine"), che genera la matrice RGB di pixel (vedi appendice B). A questo punto possiamo convertire l'immagine a colori in una in scala di grigio (con la funzione im2gray()), oppure estrarre una delle matrici dei canali di colore RGB, per poi trasformarla in immagine binaria.

Questa conversione può essere svolta in 2 modi. Il primo fissa una soglia oltre la quale convertire un pixel in un pixel bianco: dal momento che l'immagine sarà comunque ben definita (perimetro colorato su sfondo bianco), è possibile fissare una soglia alta (possibilmente pari a 255) oltre la quale trasformare un pixel in bianco, mentre quelli sotto la soglia verranno trasformati in neri. Precisamente, se I è la matrice di pixel ottenuta dai precedenti passaggi e s il valore di soglia, eseguendo il comando $BW = I > s$, si ottiene una matrice logica BW i cui elementi sono pari ad uno se i rispettivi elementi di I soddisfano la condizione I>s, altrimenti saranno pari a zero. È però preferibile che non siano gli elementi dello sfondo bianco ad essere pari ad uno, ma piuttosto quelli del perimetro, che nel caso descritto sarebbero pari a zero; per ottenere questa conversione si può usare $BW = mod(BW+1, 2)$, dove mod(,n) è la funzione modulo (resto della divisione per n, secondo argomento passato).

Un'altro metodo per ottenere l'immagine binaria è semplicemente quello di usare la funzione imbinarize(I), che trasforma I in matrice binaria scegliendo automaticamente un buon valore della soglia.

Abbiamo quindi ottenuto un'immagine (matrice binaria) con un numero fissato di pixel (numero fissato di elementi della matrice), la cui larghezza, in pixel, indichiamo con W, e l'altezza, sempre in pixel, con H. Si riconosce facilmente esistere una corrispondenza univoca tra l'immagine/matrice e la griglia da sovrapporre all'immagine per applicare il box-counting. La dimensione L del pixel (lato del quadrato che rappresenta il pixel) rappresenta il lato δ della griglia. Tuttavia, se il numero di pixel è fissato, come generare conteggi di $N(\delta)$ differenti per diversi valori di δ?

Per ottenere $N(\delta_j)$ al variare di δ_j, con $j = 1, ..., n$, si può suddivere questa matrice in n sottomatrici quadrate di dimensione diverse $dim_j \times dim_j$ e lato $\delta_j = a_j * L$, con $a_j \in \mathbb{N}$, e contare il numero $N(\delta_j)$ di sottomatrici che presentano almeno un elemento di valore 1, se si è impostato come bianco il perimetro e nero lo sfondo, o almeno uno di valore 0, se si è compiuta la scelta opposta. Una volta trovato $N(\delta_j)$ per i diversi

valori di δ_j, possiamo applicare la regressione lineare dell'equazione 3.3 per trovare la dimensione frattale D.

In ogni caso questo metodo presenta alcune difficoltà: innanzitutto si devono generare e convertire 4395 immagini, operazione non da poco. In più, immagini differenti possono essere generate da imread() con un numero diverso di pixel, e inoltre non conosciamo la precisione e il metodo con cui imread() e im2gray() e imbinarize() operano sull'immagine. In più, la condizione di avere un numero finito di pixel su cui agire può essere uno svantaggio, poichè al diminuire di dim, si raggiunge una condizione statica in cui il numero di sottomatrici intersecate dal perimetro non varia, oltre al fatto che, naturalmente, non possiamo scendere a scale minori di quella della dimensione di ciascun pixel.

4.2.2 Implementazione personale del codice di box-counting

Abbiamo optato per un secondo metodo, che lavora direttamente con le coordinate S(i).X e S(i).Y dei punti dei perimetri di ciascun ghiacciaio i-esimo, senza passare per l'elaborazione di immagini. Questo metodo ha lo svantaggio di essere meno generico, dal momento che si può applicare solo nel caso in cui l'oggetto da trattare sia curvilineo (dimensione topologica 1) e disponiamo delle coordinate dei punti che compongono l'immagine, ma più intuitivo, senza passare per le tecniche di conversione delle immagini operate dalle funzioni di Matlab. In realtà potrebbe essere esteso anche ad oggetti bi/tridimensionali: lo accenneremo nel proseguio del capitolo.

L'obiettivo è ancora quello di applicare un algoritmo di box-counting, che prevede di suddividere il grafico del perimetro (o, più in generale, di qualsiasi curva) con una griglia quadrata di definita spaziatura δ e poi di determinare il numero $N(\delta)$ di quadrati della griglia intersecati dal perimetro della figura in esame (vedi figura 4.3). Infine, conoscendo $N(\delta)$ per diversi valori di δ, applichiamo la regressione lineare 3.3 per trovare la dimensione frattale D.

Il primo passo è quello di generare una matrice binaria (matrice di soli uni e zeri), i cui elementi corrispondono ciascuno ai quadrati di lato δ della griglia sovrapposta all'immagine del ghiacciaio (vedi figura 4.3), in modo che sia possibile contare il numero di quadrati $N(\delta)$ per cui passa il perimetro del ghiacciaio (quadrati rossi in figura 4.3).

Ipotizziamo che il ghiacciaio sia racchiuso interamente in un quadrato di lato L, detto anche bounding box. La generazione della matrice è ad opera della funzione $gimbin(S(i), dim)$, scritta dall'autore (vedere appendice C.1): essa riceve in ingresso le coordinate dei punti che delimitano il ghiacciaio (contenute nella struttura $S(i)$) e il numero intero dim, per poi restituire una matrice quadrata di dimensione $dim =$

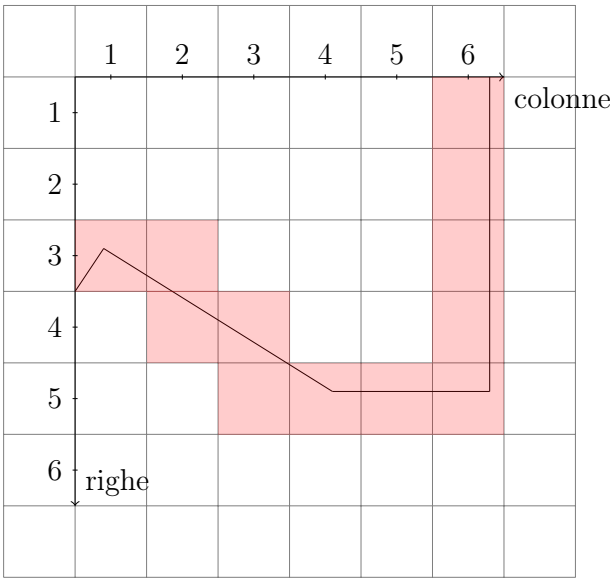


Figura 4.3: Rappresentazione della corrispondenza tra griglia sovrapposta all'immagine e matrice. Le linee nere che attraversano la griglia rappresentano un'ipotetica porzione del perimetro di un ghiacciaio. I quadrati rossi corrispondono agli elementi di matrice pari ad uno (attraversati dal ghiacciaio), mentre quelli bianchi saranno nulli (non toccati dal ghiacciaio).

L/δ [4]. La matrice descrive il ghiacciaio $T(i)$ "simulando" la griglia quadrata da sovrapporre al ghiacciaio (corrispondenza griglia⇔matrice in figura 4.3), in modo che sia possibile riconoscere per quali quadrati della griglia passa il perimetro e quindi porre uguali ad 1 gli elementi di matrice corrispondenti (quadrati rossi in figura 4.3), mentre gli altri saranno posti pari a 0 (quadrati bianchi in figura 4.3).

Sommando poi tutti gli elementi di matrice, otterremo il conteggio del numero di box $N(dim)$ occupati dal bordo dei ghiacciai. Aumentando il valore di $dim = L/\delta$ (cioè rendendo più fitta la griglia, diminuendo δ) possiamo andare a ispezionare i valori del conteggio di box-counting per diverse lunghezze δ del lato della griglia quadrata sovrapposta all'immagine.

Attenzione: quello che viene fatto è diverso dal tradizionale metodo di box-counting. Noi non generiamo l'immagine binaria ad una certa risoluzione δ per poi suddividerla in sottomatrici quadrate ed eseguire il conteggio di quali sottomatrici hanno almeno un elemento diverso da zero. Ciò che ci proponiamo di fare è generare una matrice (immagine binaria) per ogni valore $\delta(dim)$ del lato dei quadrati della griglia sovrapposta al ghiacciaio: essendo il ghiacciaio contenuto sempre nella stessa porzione rettangolare di spazio (individuata dal bounding-box), all'aumentare della dimensione dim della matrice aumenta anche la risoluzione dell'immagine binaria, cioè diminuisce il lato δ dei quadrati della griglia. Generando queste immagini per diversi valori di dim (quindi anche diversi δ), per eseguire il box-counting e trovare il numero $N(dim)$ di box occupati dal ghiacciaio, noi non abbiamo bisogno di suddividere la matrice binaria in sottomatrici e poi eseguire il conteggio di quelle non nulle, ma sommiamo semplicemente gli elementi delle singole matrici binarie generate di volta in volta con uno specifico valore di dim. Cioè, al posto di avere l'immagine binaria con la risoluzione più grande e suddividere i suoi pixel in sottomatrici di dimensioni maggiori rispetto alla risoluzione minima, noi generiamo direttamente immagini binarie diverse per ogni valore $\delta(dim)$ del lato dei quadrati della griglia ed eseguiamo il box-counting su ciascuna di queste, all'aumentare del valore di $dim = L/\delta$ (diminuire di δ).

L'idea che sta alla base del codice è quindi quella di generare una matrice e di far corrispondere ad ogni suo elemento un quadrato della griglia sovrapposta all'immagine del ghiacciaio (vedi figura 4.3). Per implementare la corrispondenza griglia-matrice possiamo determinare il box in cui è racchiuso il ghiacciaio (sfruttando il campo BoundingBox della mapstruct) e pensarlo come una matrice $(\lceil \frac{H}{\delta} \rceil) \times (\lceil \frac{W}{\delta} \rceil)$ [5], dove H è l'estensione in altezza del ghiacciaio ($H = max(S(i).Y) - min(S(i).Y) = S(i).Boundingbox(2,2) - S(i).Boundingbox(1,2)$) mentre W è l'estensione in larghezza del ghiacciaio ($W = max(S(i).X) - min(S(i).X) = S(i).Boundingbox(2,1) -$

[4] Quindi dim è il numero di quadrati della griglia lungo una sola dimensione (numero totale di quadrati pari a dim^2), o equivalentemente il numero di intervalli in cui suddividiamo il lato L del boundingbox per generare la griglia.

[5] $\lceil a \rceil$ è la parte intera superiore di a, cioè il primo intero maggiore di a.

$S(i).Boundingbox(1,1)$). In questo modo è possibile far corrispondere a ogni quadrato della griglia un solo elemento della matrice (vedi figura 4.3) e assegnargli il valore 1 se esso è attraversato dal perimetro (quadrati rossi nell'immagine 4.3) del ghiacciaio o zero altrimenti (quadrati bianchi).

In realtà è più comodo impostare le dimensioni della matrice (numero di volte in cui l'immagine viene suddivisa) piuttosto che conoscere la spaziatura δ della griglia quadrata; quindi, una volta scelta la dimensione dim da dare alla matrice, vorremmo generarne una quadrata $dim \times dim$ che ricopra il ghiacciaio come, appunto, fosse una griglia. Tuttavia il box che racchiude il ghiacciaio non è necessariamente quadrato, ma rettangolare, il che significa che dobbiamo rinunciare all'idea di costruire una matrice perfettamente quadrata. Quello che possiamo fare è determinare il $max(W,H)$ e trovare la spaziatura $\delta = max(W,H)/dim$, poi associare all'immagine la matrice $dim \times (\lceil \frac{W}{\delta} \rceil)$ se $max(W,H) = H$, mentre $(\lceil \frac{H}{\delta} \rceil \times dim)$ se $max(W,H) = W$. Questa matrice ha una delle dimensioni pari a dim e l'altra minore (cioè è $dim_1 \times dim_2$, dove $max(dim_1, dim_2) = dim$), ma è del tutto equivalente alla matrice $dim \times dim$ in termini di conteggio dei quadrati delle griglia intersecati dal profilo del ghiacciaio: infatti la parte tralasciata non può generare conteggi, dal momento che non contiene vertici del ghiacciaio.

Ora possiamo generare la matrice binaria che descrive il ghiacciaio: tendendo conto che i suoi elementi sono pari ad uno se i corrispondenti elementi della griglia sono attraversati dal perimetro del ghiacciaio, e sono nulli se non sono attraversati, per trovare $N(dim)$ sommiamo semplicemente tutti gli elementi della matrice. Ripetendo questa operazione per valori diversi di dim, $dim_1, dim_2, ..., dim_n$, e determinando i rispettivi $N(dim_i)$, con $i = 1, ..., n$, possiamo infine eseguire la regressione lineare 3.3 per trovare la dimensione frattale D, ricavando per ogni valore di dim_i il corrispettivo valore del lato dei quadrati della griglia $\delta_i = max(W,H)/dim_i$.

4.2.3 Metodo area-perimetro

Un'altra tecnica che adotteremo per determinare la dimensione frattale del perimetro dei ghiacciai è ricavata dalla relazione area-perimetro 3.6.

In realtà è possibile percorrere più di una strada: possiamo valutare il perimetro $L(\delta)$ e l'area $A(\delta)$ di ciascun ghiacciaio per diversi valori di δ (spaziatura della griglia quadrata con cui ricopriamo il ghiacciaio, vedi 3.1.3) e poi eseguire su questi una regressione lineare del tipo (ricavata dall'equazione 3.6)

$$ln(P(\delta)) = \frac{D}{2} ln(A(\delta)) + c, \tag{4.1}$$

dove $c = D * ln(\rho)$, per diversi valori di δ. In questo modo si otterrebbe un valore della dimensione frattale D per ciascuna singola figura, ma è una tecnica scomoda,

pochè determinare perimetri e aree delle figure ad una certa risoluzione δ presenta delle difficoltà.

Un'altra possibile strada è invece quella di considerare, a fissato δ, una classe di figure simili e determinare per ciascuna figura il valore di area A e perimetro P, per poi usare nuovamente la regressione lineare 4.1; in questo caso, però, i dati su cui la regressione agisce sono le aree e i perimetri delle figure singole a δ fissato, quindi non viene determinata una dimensione frattale per singola figura, ma una dimensione media della classe di figure in esame. Utilizzare figure simili equivale in un certo senso a variare il valore di δ: al posto di cambiare esplicitamente la dimensione δ^2 dei box con cui la figura viene ricoperta, riscaliamo la figura in modo che i box, a confronto della figura, risultino più piccoli o più grandi. Nel caso della classe dei ghiacciai tuttavia le figure non sono simili nel senso matematico esatto (forme identiche su scale diverse), per questo parliamo di dimensione frattale media e non del singolo ghiacciaio.

Il fatto che ρ dipenda da δ è molto scomodo per quanto riguarda la valutazione della dimensione frattale D; per questo motivo è più semplice utilizzare questo secondo metodo che analizza una classe di figure simili a valore fissato di δ. Nel caso dell'analisi dei ghiacciai nel capitolo 5, applicheremo questo metodo, dove la classe di figure in esame è quella dei ghiacciai e il valore di δ è la precisione della misurazione (quindi $A(\delta)$ e $P(\delta)$ sono i valori effettivi di area e perimetro dei ghiacciai).

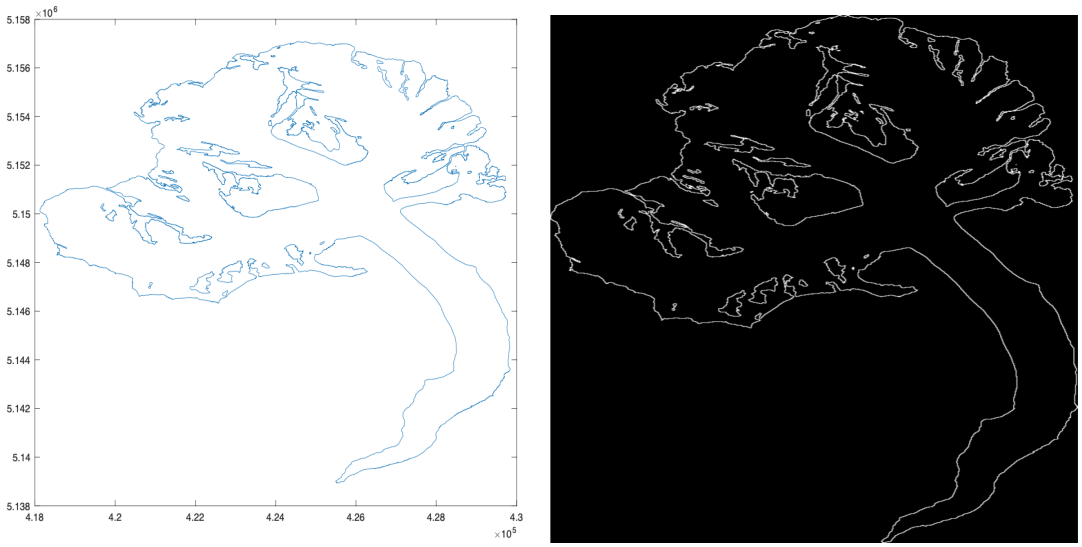
Determinare la dimensione frattale con tecniche differenti sarà utile a testare la credibilità dei risulati ottenuti.

4.2.4 Test sul codice

Abbiamo eseguito dei test sul codice per verificarne il funzionamento. Una prima verifica qualitativa viene eseguita confrontando il profilo di ghiacciai generato da plot Matlab (figura 4.4a) con l'immagine binaria ottenuta da gimbin (4.4b). I due profili tracciati sembrano uguali, anche se non possiamo essere certi della correttezza di tutti i dettagli.

Il primo insieme su cui andiamo a testare il codice per capire in che modo predice la sua dimensione frattale è un quadrato: definita la struttura-quadrato Q con i campi BoundingBox, X e Y, le abbiamo applicato la funzione $I(dim) = gimbin(Q, dim)$, per diversi valori di $dim = 2^0, 2^1, 2^2, ..., 2^{14}$ (indica il numero di volte in cui suddividiamo l'immagine e la dimensione della sua matrice binaria). Dopodichè sono state eseguite le somme sugli elementi di ciascuna matrice $I(dim) \Rightarrow N(dim) = sum(sum(I(dim)))$ e calcolati i lati $\delta(dim)$ dei quadrati di ciascuna griglia (al variare di dim), per poi usare la funzione *polyfit(log(δ(dim)), log(N(dim)), 1)* [6], la quale prende come argomento due array e il grado massimo del polinomio di fitting e restitusce il coefficiente

[6] In alternativa si può usare la funzione *fitlm(log(δ(dim)), log(N(dim)))*, che oltre a restituire i parametri della retta di regressione calcola anche le sue proprietà statistiche.



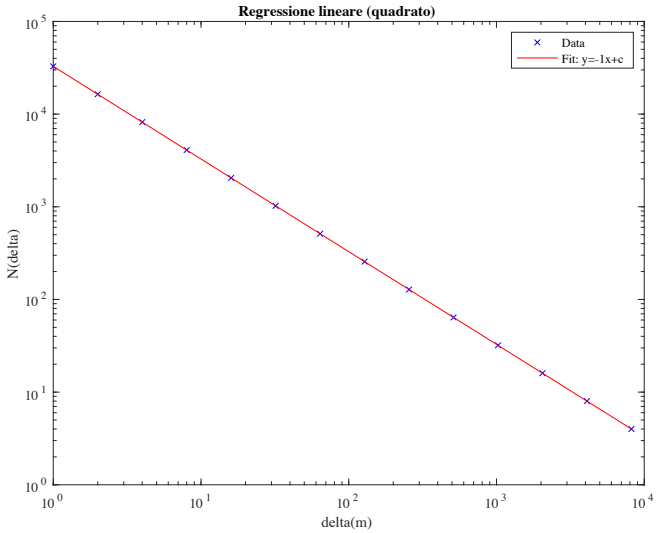
(a) Immagine a colori. (b) Immagine binaria.

Figura 4.4: Confronto di immagine 4.4a (ghiacciaio dell'Aletsch, Svizzera), generata con la funzione plot() di Matlab, e 4.4b, generata con la funzione gimbin(). L'accordo qualitativo sembra buono.

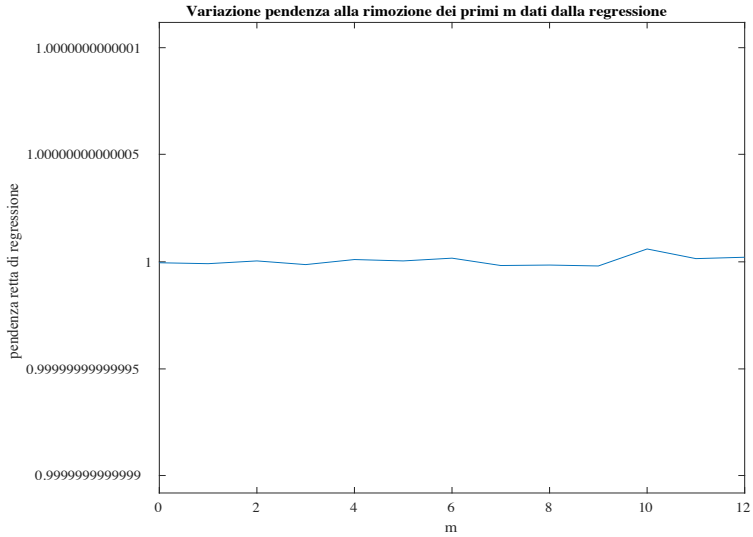
angolare e il termine noto della retta di fitting. In questo caso si riesce ad ottenere come risultato una dimensione (opposto del coefficiente angolare, vedi equazione 3.1) $D = 1$ precisa (vedi figura 4.5a). Dal momento che l'equazione 3.1, da cui nasce questa analisi, è valida per valori piccoli di $\delta \propto 1/dim$, cioè per valori grandi di dim, scegliamo anche di tracciare un grafico della variazione della pendenza della retta di regressione ottenuta eliminando via via un numero m crescente dei primi punti $(N(dim), \delta(dim))$ (cioè eliminando i punti tali che $dim = 2^0, ..., 2^m$ $m < 14$, quelli col valore più basso di dim). Il grafico 4.5b mostra che la dimensione del quadrato differisce dal valore 1 solo per cifre decimali inferiori alla quattordicesima ($|D - 1| < 10^{-14}$).

Il secondo test, invece, viene eseguito su un'insieme frattale, la polvere di Cantor (o insieme due-terzi di Cantor, vedi sezione 2.4.1). Viene generato a partire dalla funzione riportata in appendice C.3.2, che prende come argomento una struttura descrivente l'insieme di partenza della generazione (nella sezione 2.4.1 è l'intervallo unitario $[0, 1]$).

È da tenere in considerazione il fatto che, naturalmente, non possiamo generare il vero e proprio frattale, che richiederebbe un numero infinito di passaggi, ma solo un suo prefrattale di un certo ordine n. Questo però non invalida il box-counting, dal momento che, scegliendo le giuste lunghezze dei box, si possono coprire tutti i dettagli del frattale su una certa scala (definita dal lato dei quadrati della griglia, cioè



(a) Retta di regressione lineare di $(log(N(dim))$ su $log(\delta(dim))$ per l'immagine di un quadrato. Il coefficiente angolare è pari a -1, opposto della dimensione dell'insieme.



(b) Variazione della dimensione di box-counting (uguale a quella topologica) del quadrato come conseguenza della progressiva eliminazione dalla regressione dei dati con valori di dim basso.

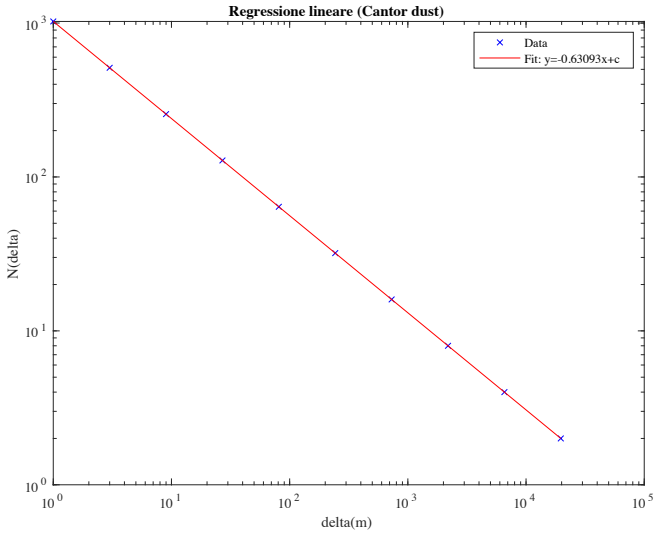
Figura 4.5

da dim) anche solo analizzando prefrattali di ordine finito. Ad esempio, generato il prefrattale n-esimo della polvere di Cantor e scelto di suddividere la sua immagine (che è semplicemente un intervallo) in 3^n parti uguali (fissato $dim = 3^n$), avendo poi trovato un numero $N(dim)$ di sottointervalli occupati dal pre-frattale, anche analizzando prefrattali di ordini $j > n$ successivi, che contengono più dettagli, si ottiene sempre un numero di box occupati pari ad $N(dim)$. Questo accade perchè, se la lunghezza del sottointervallo è a, una volta che il box-quadrato di lato $a/3^n$ contiene un punto degli estremi dei segmenti che compongono il prefrattale n-esimo C_n, lo conterrà anche se analizzassimo prefrattali di ordine maggiore[7].

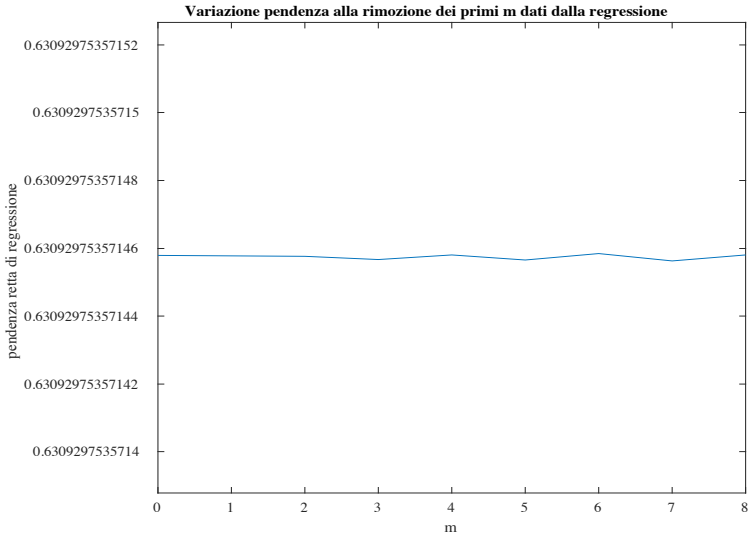
Quest'ultima osservazione ci permette di ottenere l'immagine binaria $I(dim) = gimbin(C_n, dim)$ al variare di $dim = 3^0, 3^1, 3^2, ..., 3^{10}$ generando per ogni valore di dim il corrispondente prefrattale di ordine $n = log_3 dim$, per poi eseguire il conteggio degli elementi di matrice e usare la regressione lineare, riportata in figura 4.6a. Il valore della dimensione frattale ottenuto è di $D = 0.6309297535714580$, in accordo col valore teorico $ln(2)/ln(3)$ fino alla penultima cifra decimale. È da precisare che, durante l'esecuzione del codice per questo frattale, ci siamo resi conto che i vari calcoli (in particolare le divisioni e le parti intere) usati dalla funzione gimbin() conducono ad approssimazioni che minano la correttezza del risultato. Per fare in modo di ottenere solamente risultati interi dalle divisioni, abbiamo scelto di ampliare l'intervallo generatore da $[0, 1]$ a $[0, 3^{10}]$: in questo modo, anche per il valore più grande di $dim = 3^{10}$, il lato dei box-quadrati in cui suddividiamo l'immagine ha lunghezza unitaria. Possiamo poi ripetere l'analisi della variazione della dimensione eliminando man mano i primi dati con il minor valore di dim, da 3^0 a 3^m, con $0 \leq m < n$, e dalla figura 4.6b notiamo che anche in questo caso il valore della dimensione frattale è molto preciso.

Conduciamo un'ultimo test sulla curva di Koch. Anche in questo caso suddividiamo l'immagine in parti multiple di 3 e scegliamo come generatore del frattale l'intervallo $[0, 3^{11}]$. Rispetto al test precedente si nota però che il numero di box intersecati rimane costante su una scala fissata (fissato valore di dim) solo se l'ordine della generazione n del prefrattale K_n è pari o maggiore a $n = log_3(dim*3) = log_3(dim)+1$. Fatta questa precisazione, seguendo lo stesso procedimento degli altri due test, determiniamo $I(dim) = gimbin(K_n, dim)$ al variare di $dim = 3^0, 3^1, 3^2, ..., 3^{10}$, $n = log_3(dim * 3)$ e troviamo $D = 1.279$, non in perfetto accordo col valore teorico $D = ln(4)/ln(3) = 1.2619$ (vedi figura 4.7a). Vorremmo a questo punto aumentare il numero di dati coinvolti nella regressione lineare di $log(I(dim))$ vs $log(N(dim))$, che comporterebbe generare K_{12} e valutare $N(dim)$ per $dim = 3^{11}$. Purtroppo Matlab impedisce di generare matrici binarie così grandi (corrispondenti a griglie dai box molto piccoli) e quindi non riusciamo nel nostro intento. Ciò che possiamo fare è

[7] Ricorda che ad ogni passaggio della generazione della polvere di Cantor vengono eliminati segmenti aperti e i loro estremi sono sempre presenti in tutti i prefrattali a partire dall'odine successivo a quello del passaggio che li genera.



(a) Retta di regressione lineare di $(log(N(dim))$ su $log(\delta(dim))$ per la polvere di Cantor. Il modulo del coefficiente angolare, pari alla dimensione frattale dell'insieme, risulta in ottimo accordo col valore teorico.



(b) Variazione della dimensione frattale della polvere di Cantor conseguente alla progressiva eliminazione dalla regressione dei dati con valori di dim basso.

Figura 4.6

eliminare dalla regressione i dati col valore di *dim* più basso per vedere se questo comporta un miglioramento della stima della dimensione frattale (ricordiamo che la formula 3.1 sarebbe valida per valori alti di dim, cioè per dimensioni dei box tendenti a zero): il grafico 4.7b ci conferma che il valore della dimensione effettivamente si avvicina a quello matematico di $D = ln(4)/ln(3) = 1.2619$ e varia in maniera molto più marcata rispetto ai casi precedenti. Può essere dovuto al fatto che la convergenza al valore corretto della dimensione sia più lenta per frattali in dimensioni maggiori di 1, oltre al fatto che alcuni particolari del frattale potrebbero essere sfuggiti al codice (ricordare che il frattale esatto non può essere generato, solo un suo prefrattale può esserlo).

4.2.5 Debolezze del codice

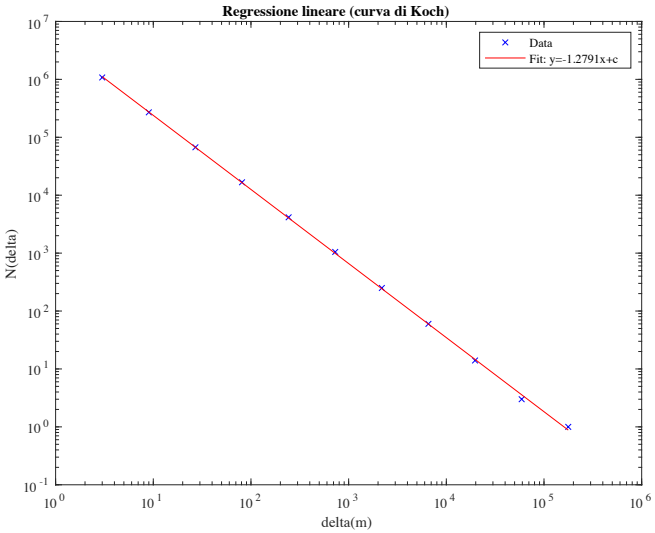
Le debolezze del codice sono già state trattate nella precedente sezione, quando abbiamo svolto i test. Raggruppiamole qui:

1. il codice si serve di una matrice binaria per eseguire il conteggio del metodo box-counting, ma questa matrice non può avere dimensioni troppo grandi (dispendio di memoria elevato). Questo problema può essere superato utilizzando la rappresentazione sparsa della matrice (sparse matrix), che, al posto di considerare una matrice come un insieme di $m \times n$ valori (m, n interi), la interpreta come una lista di indici associati al contenuto di ciascun elemento nella matrice, e tiene traccia solo di quegli indici il cui contenuto è diverso da zero, riducendo di molto il dispendio di memoria. L'approccio sparse matrix è particolarmente efficace nel caso in cui la matrice presenti una grande quantità di elementi nulli (proprio come accade nelle immagini binarie di curve).

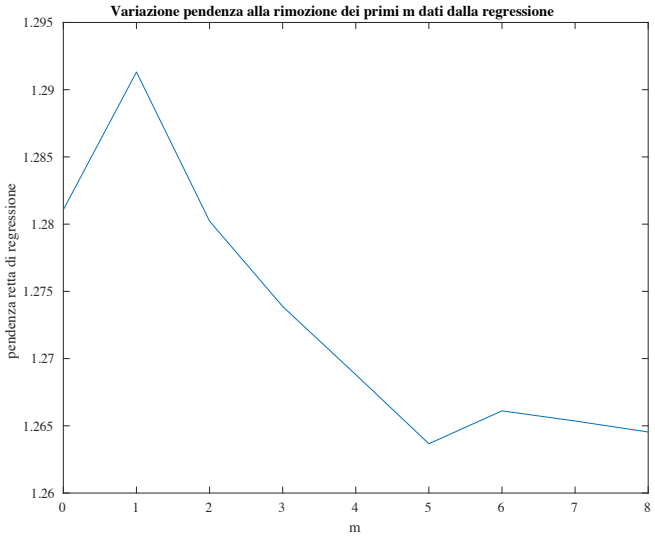
2. Le divisioni che vengono usate per determinare la lunghezza dei lati dei quadrati in cui suddividiamo l'immagine e all'interno delle parti intere (vedi codice in appendice C.1) possono causare approssimazioni fastidiose, le quali svolgono però un ruolo decisivo solo se i punti che costituiscono l'insieme frattale sono molto ordinati e privi di errore (cioè per i frattali matematici); quindi, quando tratteremo i perimetri dei ghiacciai, questo problema non si porrà.

3. Uso ristretto a insiemi di dimensione topologica 1 e costituiti da struct (possibile estensione anche ad insiemi di array). Per un'estensione del codice a superfici piane, bisognerebbe individuarne i contorni e trovare il modo di riempirli all'interno della matrice binaria. Per insiemi di dimensione topologica n maggiore, si potrebbe sfruttare il fatto che, conoscendo n proiezioni di un insieme su n piani in direzioni perpendicolari tra loro, si può ricostruire tutto l'insieme.

4. Sarebbe preferibile che il bounding-box (rettangolo contenente le coordinate del ghiacciaio) contenesse la figura al suo interno senza toccarne però i lati.



(a) Retta di regressione lineare di $(log(N(dim))$ su $log(\delta(dim))$ per la curva di Koch. Il modulo del coefficiente angolare, pari alla dimensione frattale dell'insieme, risulta in buon accordo col valore teorico.



(b) Variazione della dimensione frattale della curva di Koch in funzione della progressiva eliminazione, dalla regressione, dei dati con valori di dim basso.

Figura 4.7

Sarebbe anche preferibile fare in modo di non avere più punti successivi con coordinate x e/o y uguali e tali che stiano sui bordi più esterni delle celle che suddividono l'immagine (cioè sul bounding-box).