UNIVERSITÀ DI PISA

DIPARTIMENTO DI INGEGNERIA DELL'INFORMAZIONE

Corso di Laurea Triennale in Ingegneria Informatica

Rilevamento di curve ad S in immagini satellitari

Relatore:

Prof. Marco Cococcioni

Candidato:

Luca Ostinelli

ANNO ACCADEMICO 2023/2024

Indice

1 Introduzione alla trasformata di Hough

La trasformata di Hough è una tecnica utilizzata per rilevare forme geometriche, in particolare linee, cerchi o altri tipi di curve in un'immagine. Partiamo dal caso più semplice: la trasformata di Hough per linee. Supponiamo di avere un'immagine binaria in cui desideriamo individuare le linee presenti. La trasformata di Hough per linee si basa sull'idea che ogni linea può essere rappresentata da un punto nel parametro spazio. Questo spazio è definito dalle due coordinate $(\rho,\ \theta)$, dove ρ è la distanza dalla linea all'origine e θ è l'angolo tra la linea e l'asse x.

Il processo di trasformazione di Hough coinvolge la scansione di ciascun punto nell'immagine binaria. Per ogni punto nero, viene eseguita una scansione attraverso il parametro spazio $(\rho,\ \theta)$. Per ogni valore di ρ e θ, viene incrementato un accumulatore.

Una volta completata la scansione, i picchi nell'accumulatore indicano le linee rilevate nell'immagine. Tuttavia, è necessario impostare una soglia critica sull'accumulatore nel caso in cui si voglia selezionare solo i picchi significativi; ad esempio, rette che abbiano uno o più segmenti che insieme siano abbastanza lunghi.

La trasformata di Hough per linee è ampiamente utilizzata nelle applicazioni di visione artificiale e di elaborazione delle immagini, utili in settori come il riconoscimento di forme e oggetti, l'elaborazione di immagini mediche, la visione robotica, il controllo di qualità industriale, la guida autonoma e l'analisi di documenti.

La trasformata di Hough è stata introdotta per la prima volta da Richard Duda e Peter Hart nel 1972 (1).

Nella pagina successiva, vediamo un semplice esempio di pseudocodice per la visualizzazione di tale trasformata lineare in 3 dimensioni.

1.1 Altri tipi noti: la trasformata per i cerchi

Nel caso l'obiettivo sia trovare cerchi all'interno di un'immagine, organizziamo lo spazio in questo modo. Ogni pixel dell'immagine viene rappresentato come un punto nello spazio dei parametri Hough, dove i parametri sono le coordinate del centro $(x,\ y)$ e il raggio r. Utilizzando un accumulatore tridimensionale, vengono accumulati i voti per ogni possibile centro del cerchio e raggio, consentendo l'individuazione dei cerchi mediante l'identificazione dei picchi nell'accumulatore.

Una volta individuati i picchi nell'accumulatore, che rappresentano i centri e i raggi dei cerchi rilevati, è possibile applicare ulteriori criteri di filtraggio, come la soglia o altri criteri di selezione dei picchi, per raffinare la selezione dei cerchi.

Listing 1: Codice MATLAB per la trasformata di Hough lineare

```
imageSize = [400, 400];
img = ones(imageSize);
img_binary = zeros(imageSize);

% Primo subplot
subplot(1,2,1);
imshow(img);
hold on;
for i = 1:NumSegmenti
    % Genera casualmente gil estremi del segmento
    x1 = randi(imageSize(2)); y1 = randi(imageSize(1));
    x2 = randi(imageSize(2)); y2 = randi(imageSize(1));

    % Disegna il segmento sull'immagine
    line([x1 x2], [y1 y2], 'Color', 'black',
                           'LineWidth', lineWidth);
    img_binary = insertShape(img_binary,
                         'Line', [x1 y1 x2 y2], 'Color',
                         'white', 'LineWidth', lineWidth);
end
hold off;

img_binary = imbinarize(rgb2gray(img_binary));

% Calcola la trasformata di Hough
[H,theta,rho] = hough(img_binary);

% Calcola le coordinate y
y = 1:imageSize(1);

% Calcola le coordinate theta e y_grid
theta_deg = theta - 90; theta_rad = deg2rad(theta_deg);
y_grid = linspace(1, imageSize(1), size(H, 1));
theta_grid = linspace(theta_rad(1), theta_rad(end), size(
    H, 2));

% Secondo subplot
subplot(1,2,2);
surf(theta_grid, y_grid, H, 'EdgeColor', 'none');
```

1.2 Obiettivo

L'obiettivo di questo lavoro è sviluppare un approccio innovativo per identificare e tracciare curve a forma di S nelle immagini satellitari. Per raggiungere questo scopo, abbiamo per prima cosa parametrizzato un prototipo di curva ad S, nello specifico una sigmoide, tramite due variabili.

Dopodiché, mediante questa parametrizzazione, con la semplice applicazione dei principi standard, siamo arrivati a costruire un tensore[1] in quattro dimensioni, due che indicassero le coordinate del centro e due per la curva vera e propria, il quale conterrà il numero di occorrenze che tale curva avrebbe piazzando il centro in quel punto ha nell'immagine.

Successivamente, abbiamo modificato alcuni principi originali della trasformata per adattarla ad esigenze diverse. Tra i risultati della trasposizione della trasformata di Hough, infatti, c'è il limite abbastanza gravoso della crescita dimensionale.

La nostra metodologia, di conseguenza, si distingue perché utilizzeremo una trasformata che converte uno spazio in origine a cinque dimensioni in uno a tre dimensioni, consentendoci di tralasciare dettagli più complessi ma spesso irrilevanti e di gestire più comodamente la visualizzazione del risultato.

1.3 Trasformata generalizzata

Esiste anche un altro metodo, che enunceremo ma non tratteremo, per approcciarsi al problema. Questo tralascia l'impostazione analitica per concentrarsi sui modelli di adattamento.

Il Generalized Hough Transform (GHT)(3), introdotto da Dana H. Ballard nel 1981, è una modifica della trasformata di Hough che utilizza il principio del riconoscimento dei modelli. Mentre la trasformata di Hough originale era utilizzata per rilevare forme analiticamente definite (come linee, cerchi, ellissi, ecc.), il GHT consente di rilevare oggetti arbitrari descritti tramite modelli. Questa modifica trasforma il problema di individuare un oggetto in un'immagine nel problema di trovare i parametri di trasformazione che mappano il modello nell'immagine.

[1] la nozione di tensore generalizza tutte le strutture definite usualmente in algebra lineare a partire da un singolo spazio vettoriale(2).

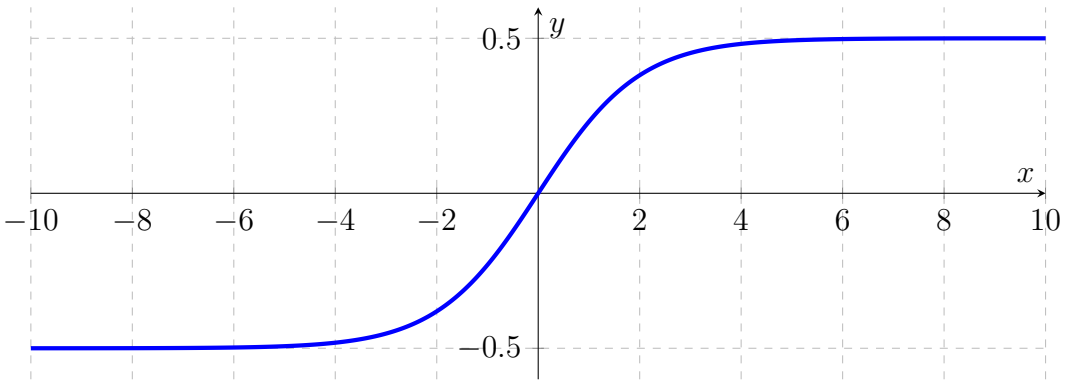
2 L'algoritmo di Hough per curve ad S

2.1 Parametrizzazione della curva

Come anticipato nell'introduzione, la prima cosa da fare è determinare quale forma analitica vogliamo andare ad individuare nell'immagine.

Prendiamo una sigmoide semplice centrata in $(0,\ 0)$:

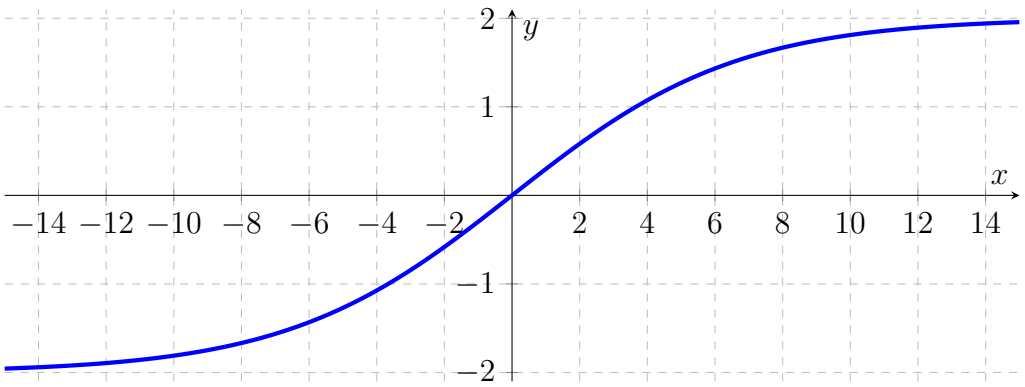
$$y(x) = \frac{e^x}{1 + e^x} - 0.5$$



Questa curva si adatta perfettamente alle necessità che abbiamo. Adesso dobbiamo gestirne la velocità con cui passa dall'asintoto inferiore a quello superiore e l'ampiezza, in modo da poter generare tutte le curve di interesse.

Il primo parametro lo chiamiamo a e il secondo d, adesso la formula diventa:

$$y(x) = d \cdot \left(\frac{e^{a \cdot x}}{1 + e^{a \cdot x}} - 0.5 \right)$$



Esempio particolare con $a = 0.3$ e $d = 4$

2.2 Trasposizione dell'algoritmo

L'algoritmo a pagina seguente, genera tutte le curve con parametri a e d in range prestabiliti. Data la complessità computazionale dell'algoritmo può avere senso, dove è concesso, limitare i domini di a e d per generare e quindi testare meno curve. Ad esempio, se volessimo solo curve con derivata prima positiva, basterebbe mantenere $a > 0$.

Una volta generata una curva, che chiameremo S, sovrapponiamo, per ogni pixel di riga c, e di colonna b, la curva sull'immagine in modo che il suo centro coincida con il pixel.

A questo punto contiamo le occorrenze dei pixel neri in corrispondenza della curva, questo numero fornisce una misura della presenza della curva di centro $(c,\ b)$.

2.3 Punti critici di questo approccio

Il porting dell'algoritmo di Hough ci mette nelle condizioni di avere un tensore di accumulazione in quattro dimensioni. Questo è un problema poiché è molto complicato trovare un modo di rappresentare le cinque dimensioni in modo comprensibile all'utente.

Disegnare un grafico in cinque dimensioni, rappresenta una sfida significativa a causa della complessità aggiuntiva nell'interpretazione visiva di più variabili. Mentre la rappresentazione di dati in due o tre dimensioni è relativamente intuitiva, aggiungere dimensioni comporta un aumento esponenziale della complessità percettiva.

La principale difficoltà risiede, quindi, nel trovare un metodo efficace per visualizzare in modo chiaro e comprensibile tutte e cinque le dimensioni simultaneamente. Le tradizionali tecniche di visualizzazione, come grafici a dispersione o grafici a barre, diventano rapidamente inefficienti e ingombranti con un numero così elevato di dimensioni. Altresì, la sovrapposizione di più informazioni sullo stesso grafico può portare a confusione e difficoltà nel distinguere tra i diversi elementi.

Un altro aspetto da tenere in considerazione è che, la memorizzazione di grandi quantità di dati in tensori, diventa computazionalmente più complesso man mano che le dimensioni dei dati aumentano. Questo perché l'aumento delle dimensioni comporta una crescita esponenziale del numero totale di elementi nel tensore.

Di conseguenza, viene limitata la capacità di elaborare immagini molto risolute, che a parità di dimensioni del tensore hanno molti più valori da salvare ed elaborare.

Listing 2: Pseudocodice per la trasformata di Hough per curve ad S

```
[righe, colonne] = size(image);

% Struttura
occorrenze = zeros(num_passi_c, num_passi_b,
                   num_passi_a, num_passi_d);

% Parametri della curva
for a_idx = min_a:passo_a:max_a
    for d_idx = min_d:max_d
        % Calcola tutte le y in funzione di x
        y = zeros(1,2*colonne+1);
        for x = -colonne:colonne
            y(x+colonne+1) = round(-d_idx*((
            (exp(a_idx*x))/(1+exp(a_idx*x)))-0.5));
        end

        % Prova la curva generata per ogni pixel
        for b_idx = 1:num_passi_b
            for c_idx = 1:num_passi_c

                contatore = 0;

                for x = 1:colonne
                    riga = c_idx+y(x-b_idx+colonne+1);
                    if riga > 0 && riga <= righe
                        if image(riga, x) == 0
                            contatore = contatore + 1;
                        end
                    end
                end
                occorrenze(c_idx, b_idx, a_idx, d_idx) =
                        contatore;
            end
        end
    end
end
```

3 Proviamo a fissare una dimensione

Come abbiamo visto in precedenza, l'approccio di Hough prevede un tensore a quattro dimensioni. Per aggirare parzialmente questo problema, possiamo fissare un parametro della curva, come ad esempio d (l'ampiezza), in modo tale da ridurre a tre il numero delle dimensioni del tensore.

Come si può osservare nello pseudocodice a pagina seguente, questa soluzione è abbastanza semplice da implementare e permette di risolvere alcuni problemi.

In primo luogo, risolviamo parzialmente il problema relativo all'impiego di memoria, poiché abbiamo diminuito il numero di dimensioni, quindi abbiamo sensibilmente meno valori da memorizzare.

Per quanto riguarda la difficoltà di rendere intelleggibile una struttura del genere, ci sono degli escamotage che permettono la visualizzazione in maniera chiara delle quattro dimensioni, facendo uso dei colori come nell'esempio seguente.

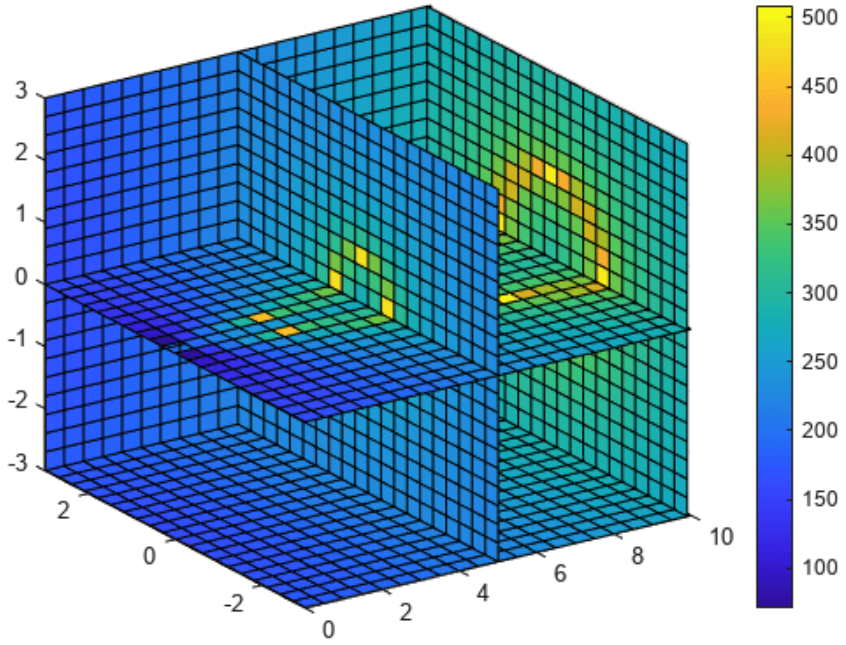


Figura 1: Esempio di Visualizzazione 4D da MathWorks (4)

Listing 3: Pseudocodice per la trasformata di Hough per curve ad S con d fissato

```
[righe, colonne] = size(image);

d_idx = valore_stimato;

% Struttura
occorrenze = zeros(num_passi_c,
                   num_passi_b,
                   num_passi_a);

% Parametro della curva
for a_idx = min_a:passo_a:max_a
    % Calcola tutte le y in funzione di x
    y = zeros(1,2*colonne+1);
    for x = -colonne:colonne
        y(x+colonne+1) = round(-d_idx*(((exp(a_idx*x))
                                /(1+exp(a_idx*x)))-0.5));
    end

    % Parametri del punto
    for b_idx = 1:num_passi_b
        for c_idx = 1:num_passi_c

            contatore = 0;

            for x = 1:colonne
                riga = c_idx+y(x-b_idx+colonne+1);
                if riga > 0 && riga <= righe
                    if image(riga, x) == 0
                        contatore = contatore + 1;
                    end
                end
            end
            occorrenze(c_idx, b_idx, a_idx/passo_a) =
                    contatore;
        end
    end
end
```

3.1 Analizziamo qualche esempio

Come accennato prima, per analizzare gli esempi, usiamo una struttura in cui la quarta dimensione è rappresentata dal colore.

Per rendere comprensibile tale struttura usiamo tre piani di taglio, uno per ogni dimensione del tensore, per andare a verificare i colori all'interno della zona d'interesse: c, b e a, rappresentati rispettivamente da y, x e z.

Nel primo esempio è possibile vedere come la curva venga identificata al centro. Tramite il punto di vista isometrico, notiamo che grazie al parametro a, si crea, sul piano verticale, una sorta di parabola che identifica la presenza di asintoti abbastanza lunghi.

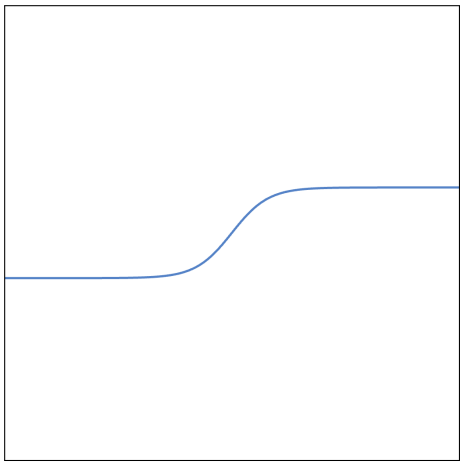
Infatti, nel secondo esempio, dove la stessa curva ha gli asintoti tagliati, notiamo una parabola sul piano verticale più stretta.

Nel terzo e quarto esempio, notiamo come sia precisa l'identificazione della posizione del centro della curva. I piani infatti sono stati collocati esattamente nel punto di massimo assoluto del tensore.

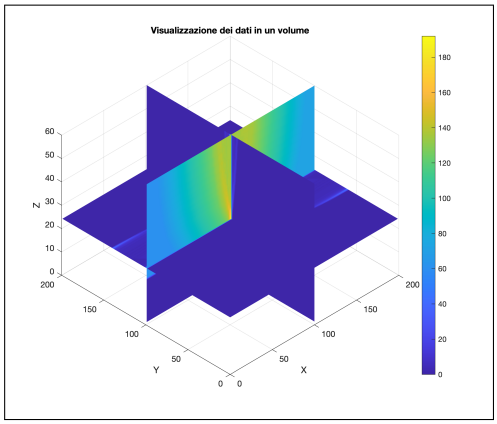
Nel quinto esempio, invece, è possibile osservare la maggior larghezza della parabola sull'asse verticale, dovuta all'importante ampiezza della curva stessa. Infatti la curva va a posizionarsi parzialmente su una serie di parametrizzazioni al variare di a.

Infine, il sesto esempio verifica la presenza di entrambe le curve singolarmente. Per comodità di lettura, gli assi sono stati posizionati su una delle due sigmoidi. Dopo vedremo che questa precisione sia dovuta al fatto che il parametro d sia fissato.

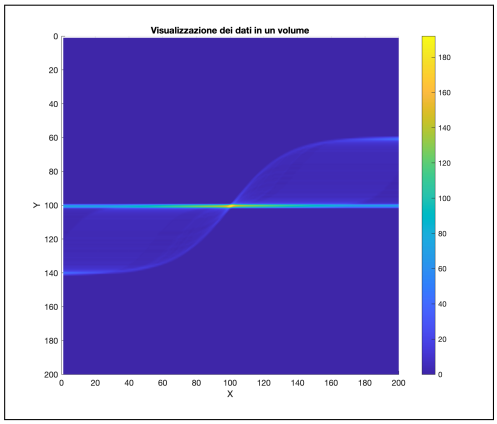
3.1.1 Esempio in 4 dimensioni n.1



Input

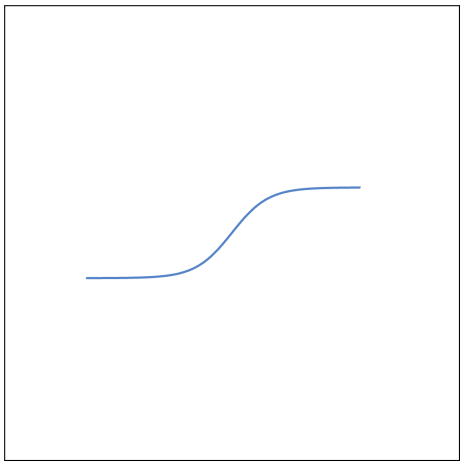


Isometrica

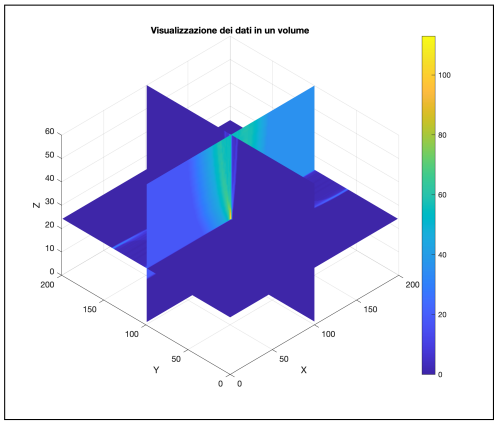


Sotto

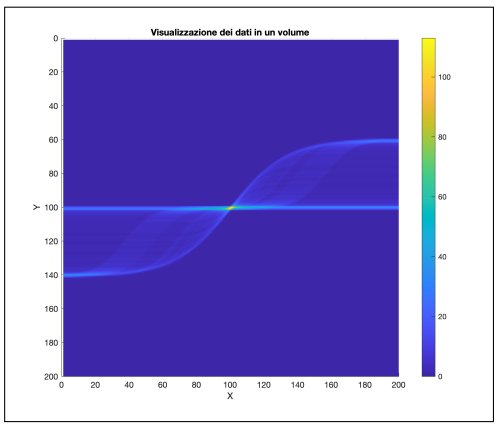
3.1.2 Esempio in 4 dimensioni n.2



Input

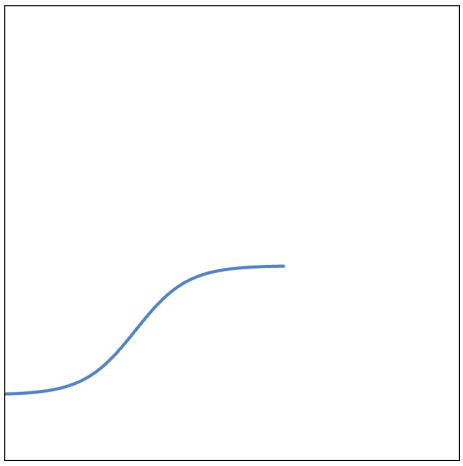


Isometrica

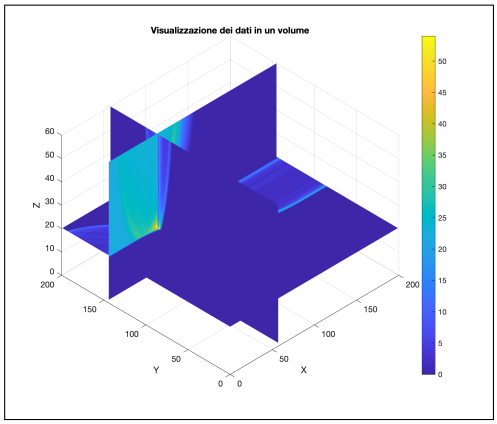


Sotto

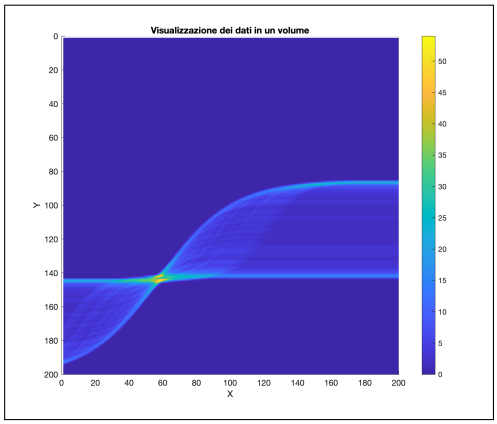
3.1.3 Esempio in 4 dimensioni n.3



Input

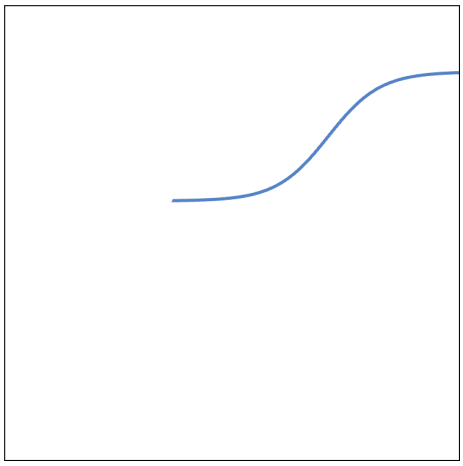


Isometrica

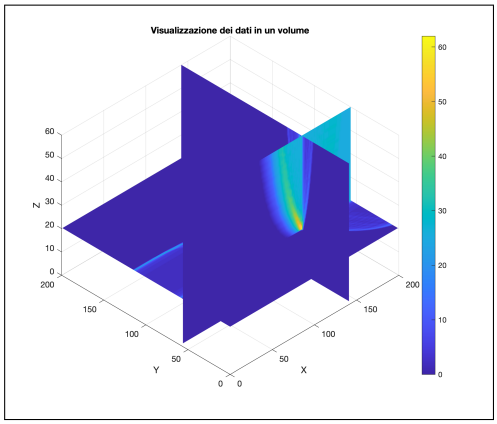


Sotto

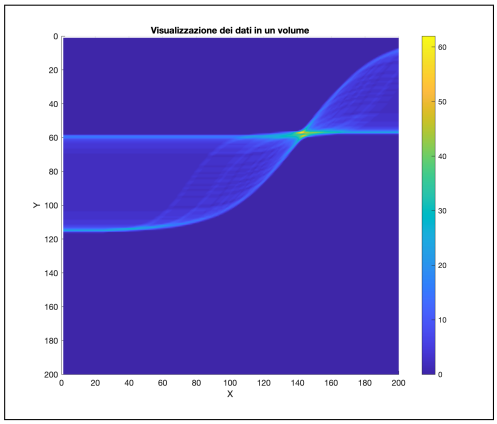
3.1.4 Esempio in 4 dimensioni n.4



Input

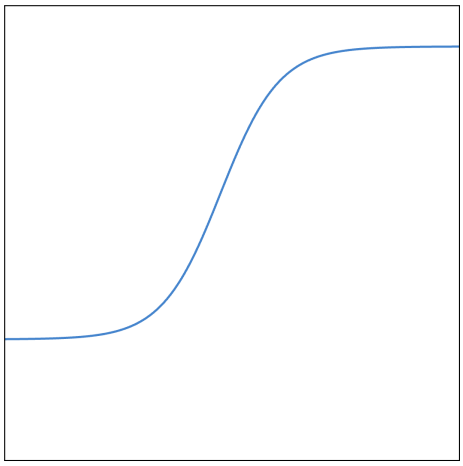


Isometrica

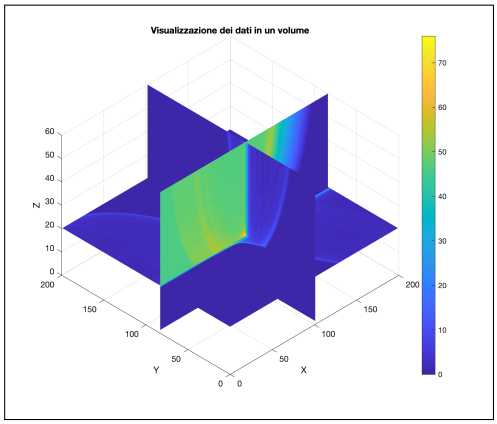


Sotto

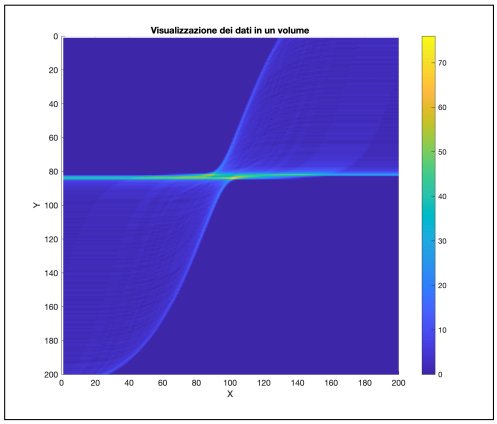
3.1.5 Esempio in 4 dimensioni n.5



Input

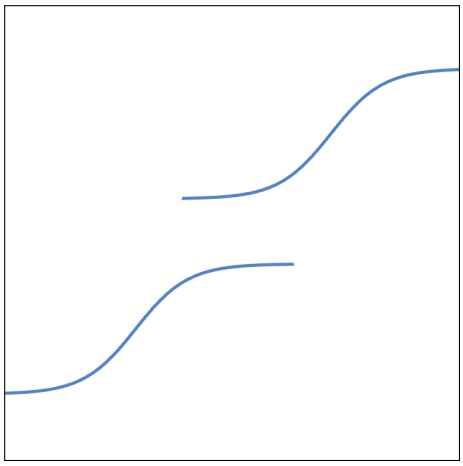


Isometrica

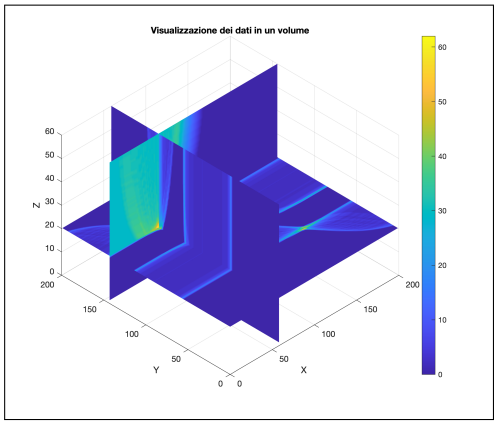


Sotto

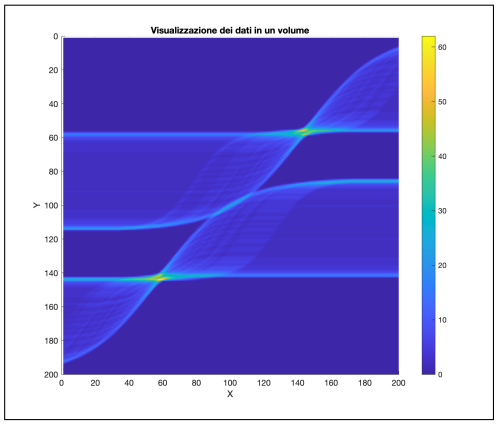
3.1.6 Esempio in 4 dimensioni n.6



Input



Isometrica



Sotto

4 Un nuovo approccio

Il nostro approccio si basa, come l'originale, sul trovare tutte le curve nei range di parametri prestabiliti per ogni pixel dell'immagine.

Quindi, a differenza del metodo visto prima, non andiamo a limitare la ricerca delle curve in qualche dimensione. Tuttavia cerchiamo di dimostrare come, nella maggior parte dei casi, sia più sensato identificare le curve in base alla posizione del loro centro rispetto agli altri due parametri.

Questo lo facciamo utilizzando una matrice di coordinate $(c,\ b)$ in cui salviamo le informazioni relative ai parametri (a e d) e il numero di occorrenze che tale curva, la migliore, ha nel punto.

Com'è possibile osservare nello pseudocodice a pagina seguente, alla riga 31 e seguenti, i dati (parametri della curva e occorrenze) vengono salvati solo nel caso in cui non ci sia stata una curva, precedentemente testata, che ha fatto un risultato migliore in termini di occorrenze in quel punto.

Questo ridurre le dimensioni del tensore a due (le coordinate del pixel) ci permette, oltre il notevole risparmio di spazio all'interno della memoria, di rappresentare il risultato della trasformata all'interno di un grafico tridimensionale, esattamente come nella trasformata lineare.

Gli assi del grafico sono x e y per le coordinate dei pixel e z per le occorrenze.

Per un essere umano è estremamente intuitivo, a questo punto, identificare il centro della curva. Anche il sistema però è in grado, con uno sforzo computazionale ridotto, di trovare il massimo locale.

Rimane il problema della soglia critica, da scegliere in base al tipo di applicazione che vogliamo fare dell'algoritmo, che è intrinseco a questo tipo di trasformate.

Nella prossima sezione, analizzeremo qualche esempio. Per ognuno di questo abbiamo a disposizione quattro visualizzazioni che riescono a rendere un'idea del grafico tridimensionale:

1. una isometrica, che permette di vedere il picco tridimensionale;
2. una da sotto, utile grazie al colore, per vedere il centro della curva;
3. vista dall'asse X;
4. vista dall'asse Y.

Listing 4: Pseudocodice per la trasformata in 2+1 dimensioni

```
[righe, colonne] = size(image);

% Strutture per occorrenze e parametri della curva
occorrenze = zeros(num_steps_c, num_steps_b);
valoreA = zeros(num_steps_c, num_steps_b);
valoreD = zeros(num_steps_c, num_steps_b);

% Parametri della curva
for a_idx = min_a:passo_a:max_a
    for d_idx = min_d:max_d
        % Calcolo tutte le y in funzione di x
        y = zeros(1,2*colonne+1);
        for x = -colonne:colonne
            y(x+colonne+1) = round(-d_idx*(((exp(a_idx*x)
               )/(1+exp(a_idx*x)))-0.5));
        end
        % Provo la curva generata per ogni pixel
        for b_idx = 1:num_steps_b
            for c_idx = 1:num_steps_c

                contatore = 0;

                for x = 1:colonne
                    riga = c_idx+y(x-b_idx+colonne+1);
                    if riga > 0 && riga <= righe
                        if image(riga, x) == 0
                            contatore = contatore + 1;
                        end
                    end
                end

                if contatore > occorrenze(c_idx, b_idx)
                    occorrenze(c_idx, b_idx) = contatore;
                    valoreA(c_idx, b_idx) = a_idx;
                    valoreD(c_idx, b_idx) = d_idx;
                end

            end
        end
    end
end
```

4.1 Analisi degli esempi

Nel primo esempio, vediamo un picco al centro del grafico che segnala la presenza del centro della curva. Nella visione dall'asse x, è possibile notare come ci siano due pendii ai lati del picco.

Questo è dovuto alla lunghezza degli asintoti: alcune delle curve centrate nello stesso punto e con parametro d (ampiezza) identico ma con parametro della curva a diverso avranno in comune quella serie di punti.

Di conseguenza, la stessa curva, presente nel secondo esempio, che tuttavia ha gli asintoti tagliati, avrà un picco maggiormente isolato senza lasciarsi la scia sui versanti.

Nel terzo e quarto esempio, è possibile notare come sia precisa la posizione di identificazione. Con una forma, apprezzabile dalle visualizzazioni ortogonali agli assi x e y, molto simile a quello del secondo esempio.

Il quinto esempio dimostra come, in questo tipo di trasformata, dove i parametri della curva vanno in secondo piano rispetto alla presenza o meno della stessa, la sigmoide presente viene segnalata ma non è apprezzabile una significativa differenza per la parte inerente ai parametri a e d.

Un caso decisamente interessante è il sesto esempio che, grazie alla presenza di due curve, palesa un fatto interessante.

I picchi dei relativi centri sono assolutamente ben visibili, di conseguenza ben individuabili anche grazie ad una soglia critica impostabile.

Da notare qualche picco più piccolo, ben visibile dall'asse x, anomalo. Questo è dovuto alla presenza, nello stesso grafico, di tutti i test delle possibili curve di parametri a e d. Di conseguenza, vengono generate curve che intersecano le due presenti nell'immagine come si vede nell'immagine sotto.

Infatti, la curva tratteggiata, potrebbe non essere presente ma essere rilevata in quanto alcuni punti andranno a votare proprio per quella.

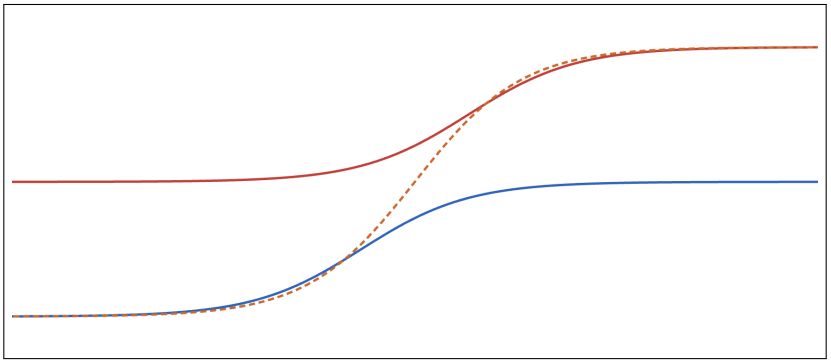
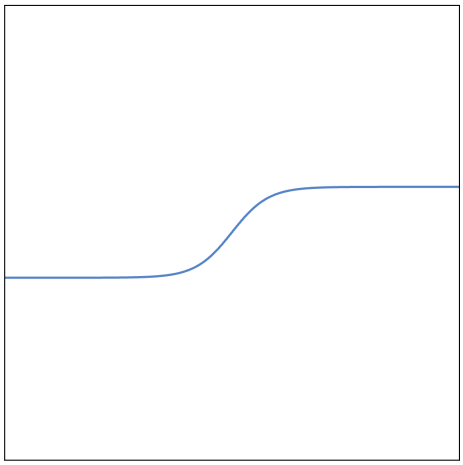
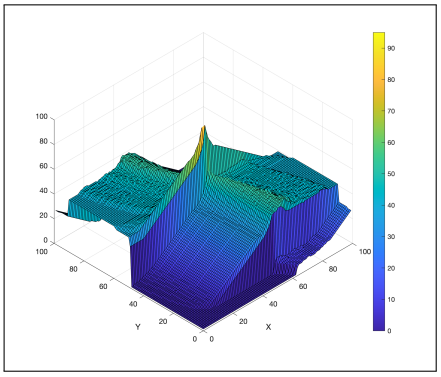


Figura 8: Esempio di curve sovrapposte con parametro d differente

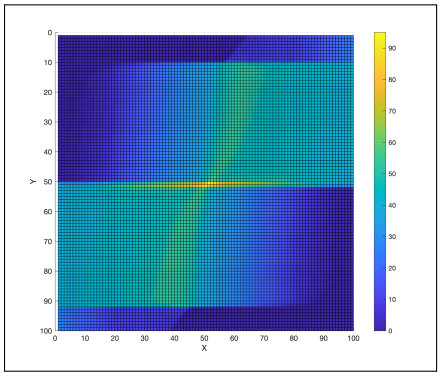
4.1.1 Esempio in 3 dimensioni n.1



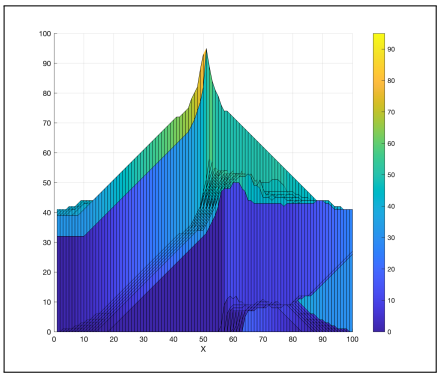
Input



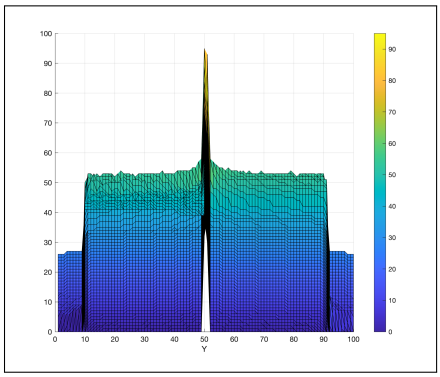
Isometrica



Sotto

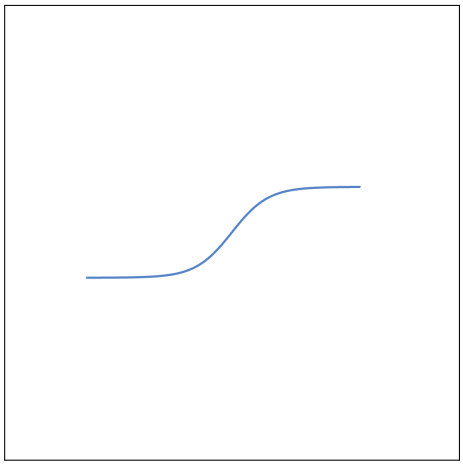


Vista da X

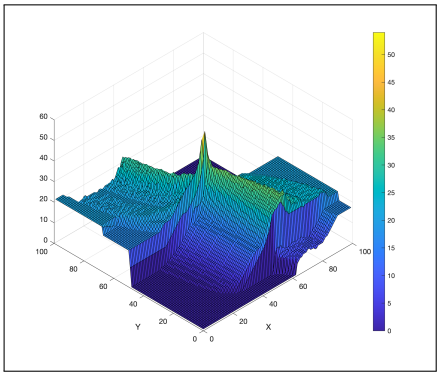


Vista da Y

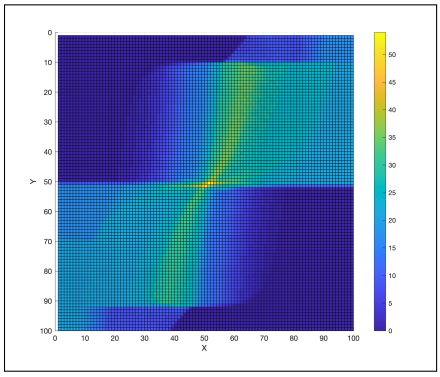
4.1.2 Esempio in 3 dimensioni n.2



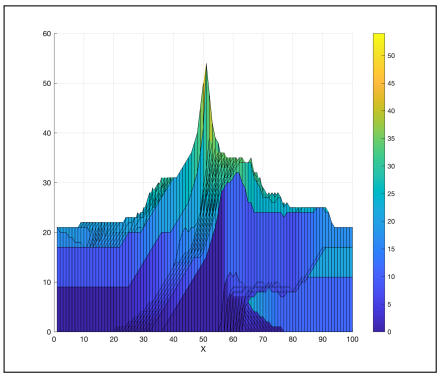
Input



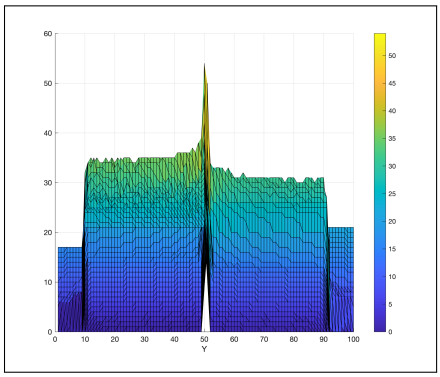
Isometrica



Sotto

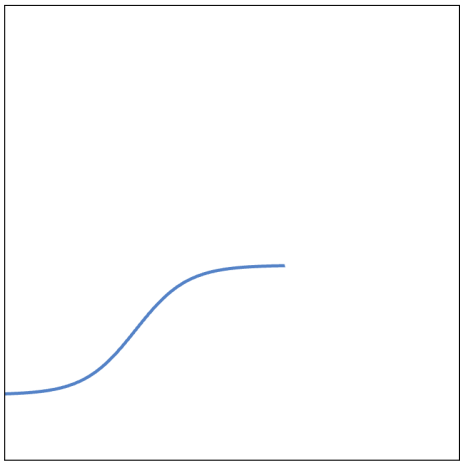


Vista da X

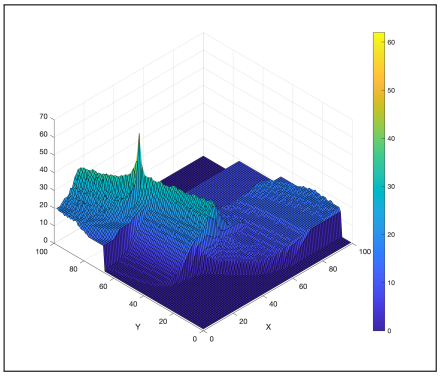


Vista da Y

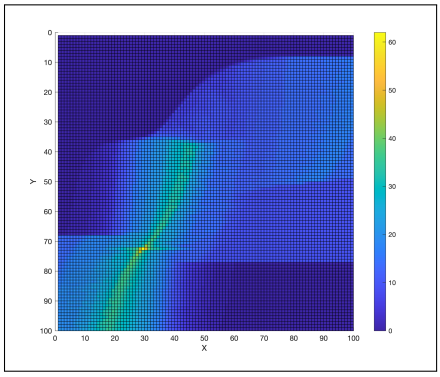
4.1.3 Esempio in 3 dimensioni n.3



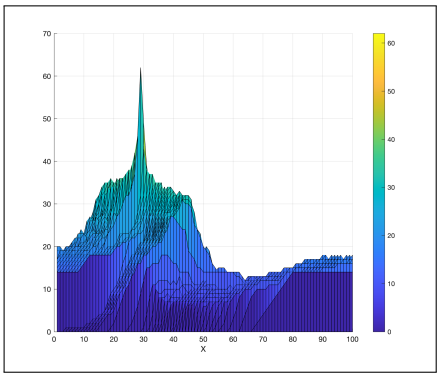
Input



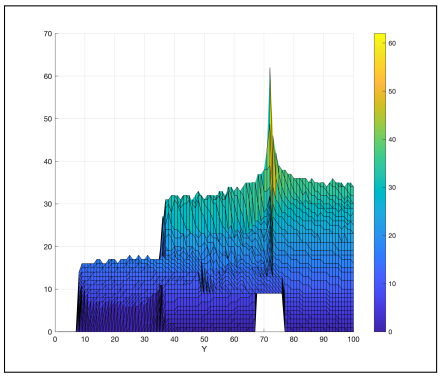
Isometrica



Sotto

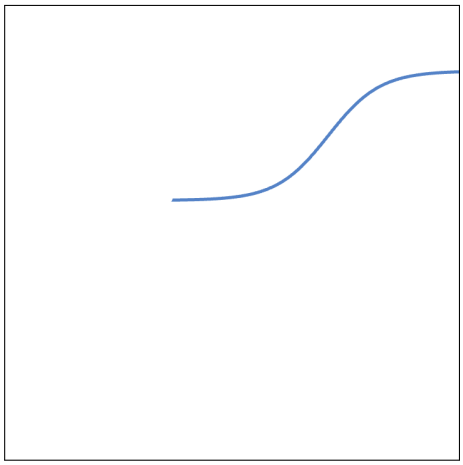


Vista da X

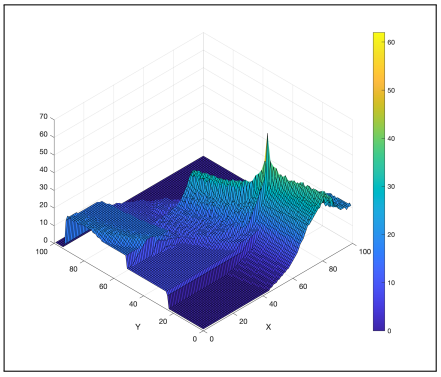


Vista da Y

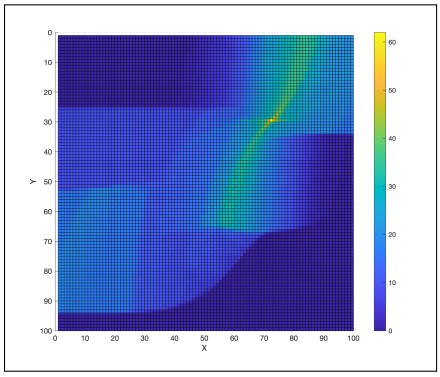
4.1.4 Esempio in 3 dimensioni n.4



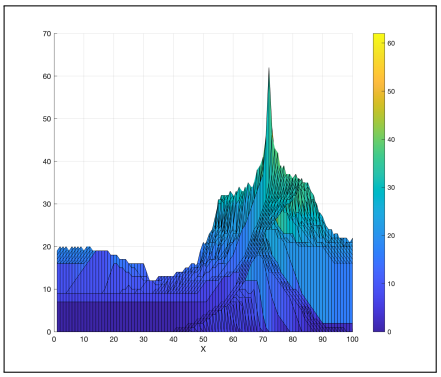
Input



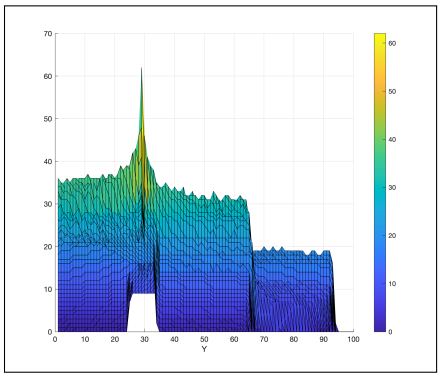
Isometrica



Sotto

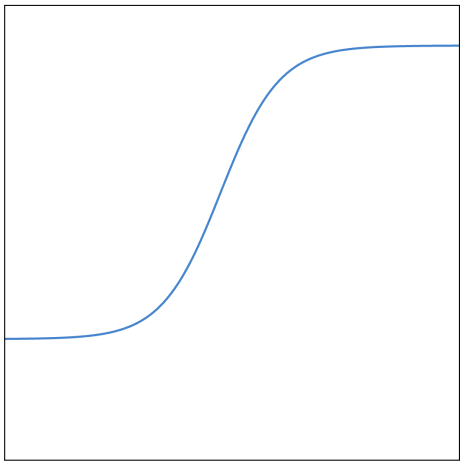


Vista da X

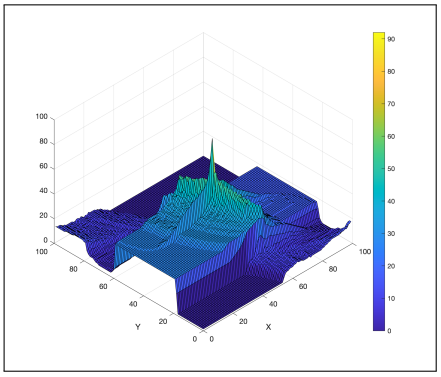


Vista da Y

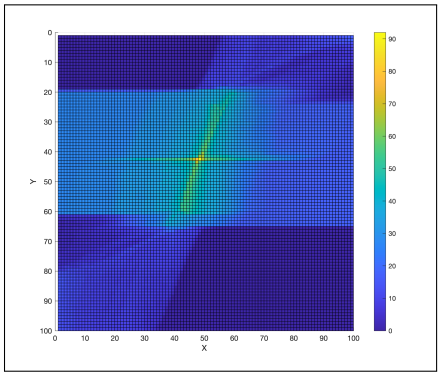
4.1.5 Esempio in 3 dimensioni n.5



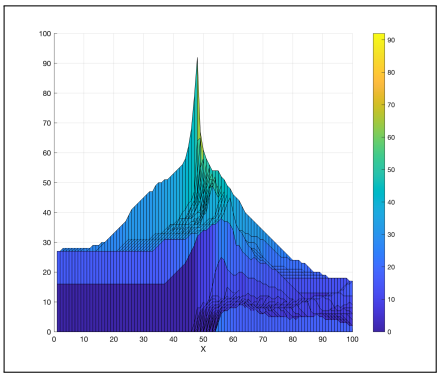
Input



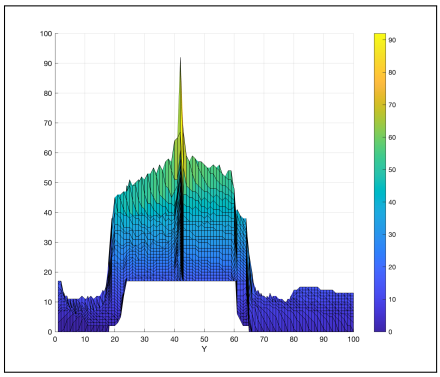
Isometrica



Sotto

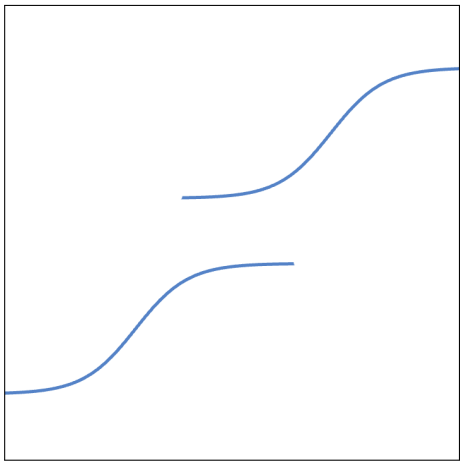


Vista da X

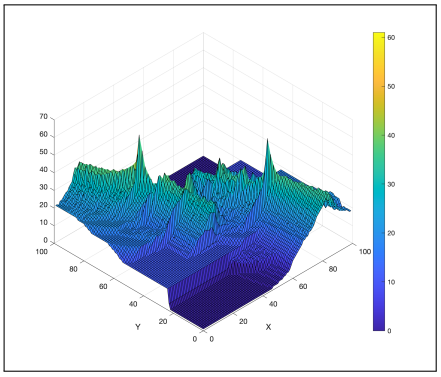


Vista da Y

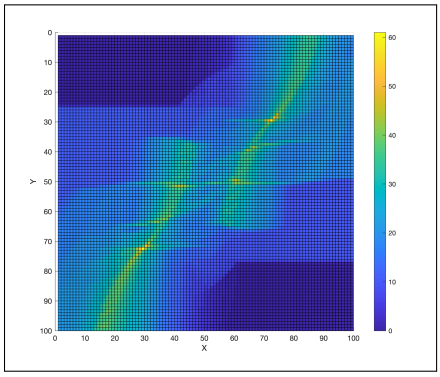
4.1.6 Esempio in 3 dimensioni n.6



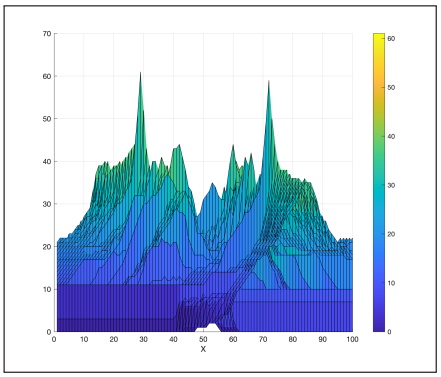
Input



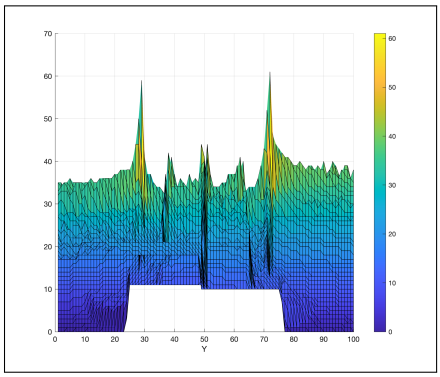
Isometrica



Sotto



Vista da X



Vista da Y

5 Proviamo con delle immagini satellitari

Le immagini satellitari sono realizzate utilizzando sensori in orbita attorno alla Terra. Questi sensori catturano la luce riflessa dalla superficie terrestre in diverse bande dello spettro elettromagnetico.

Le curve ad S segnalano la presenza di un target, ma oltre a quel segnale abbiamo un notevole rumore sottoforma di bande verticali multicolore che possono essere il risultato di riflessi atmosferici, errori di calibrazione o disturbi durante la trasmissione dei dati.

5.1 Eliminazione del rumore

Prima di applicare l'algoritmo per la generazione di grafici tridimensionali, è cruciale affrontare l'aspetto relativo all'elaborazione dell'immagine satellitare, che come detto prima presenta anche la complessità di separare il segnale dal rumore.

Per far ciò, è indispensabile sviluppare un metodo efficace per gestire questo tipo di input. Negli esempi a nostra disposizione, siamo riusciti a svolgere un discreto lavoro di separazione delle due componenti tramite un semplice filtro composto da una maschera che seleziona i toni del rosso, eliminando gli altri. Spesso potrebbe essere necessario un procedimento più complesso.

Listing 5: Codice per l'isolamento della scala cromatica d'interesse

```
redMask = uint8(img(:,:,1) > 190 & % Rosso
                img(:,:,2) < 180 & % Verde
                img(:,:,3) < 140); % Blu
img = rgb2gray(img);
img = img .* redMask;
img = ~imbinarize(img);
```

A questo punto, l'immagine ottenuta può essere passata allo stesso algoritmo di prima per ottenere la matrice delle occorrenze da graficare.

Negli esempi sottostanti, è possibile vedere anche l'esito dell'applicazione di questo filtro.

5.2 Analisi degli esempi reali

Come vediamo dagli esempi, l'algoritmo è in grado di rilevare correttamente il primo caso, che è segnalato da un picco proprio in concomitanza del centro della curva.

Per quanto riguarda il secondo caso, invece, abbiamo più considerazioni da fare.

Proprio quest'ultimo, infatti, sottopone ad un notevole stress il nostro algoritmo poiché ha una serie di curve, con parametro a, che ricordiamo essere la velocità di crescita, molto vario e un parametro d, l'ampiezza della curva, pressoché fissa.

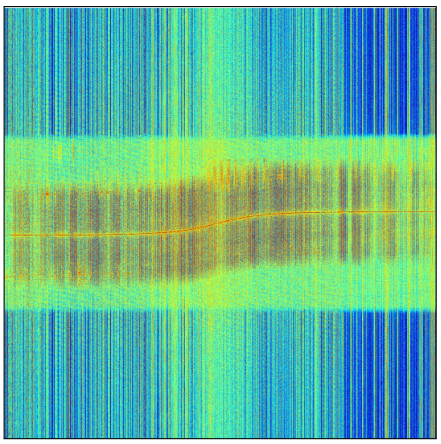
Il riconoscimento evidenzia infatti un picco in posizione della curva di maggior lunghezza, appartenente alla curva col maggior numero di occorrenze e una serie di picchi più piccoli in posizione delle altre. Inoltre l'immagine ha delle curve che sono presenti a tratti, proprio in questi casi la scelta della soglia critica ha bisogno di particolari valutazioni, in quanto è autoevidente che tali curve avranno picchi mutiliati.

Il problema che avevamo riscontrato parzialemente nel sesto esempio del capitolo precedente[2], invece, è qui di nuovo presente. Alcune interazioni tra le curve tendono a presentare picchi di minor entità.

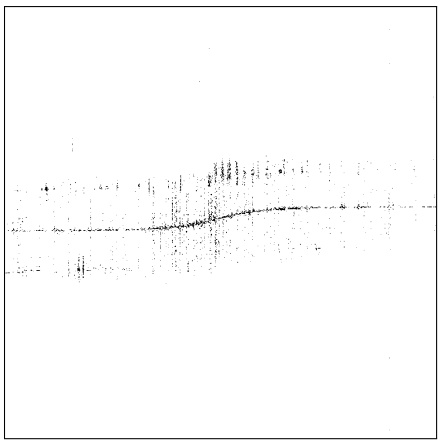
In immagini reali molto complesse, il nostro algoritmo potrebbe quindi presentare dei falsi positivi difficili da rimuovere.

[2]quello inerente a curve distinte che, per alcuni parametri a e d, potrebbero andare a votare per un'unico punto.

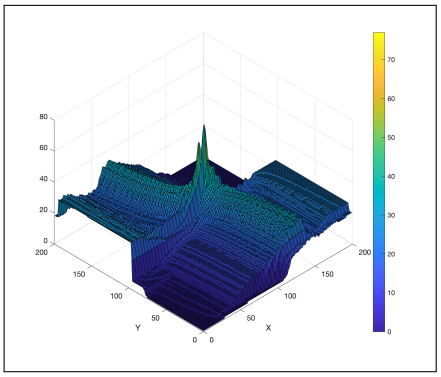
5.2.1 Esempio con immagine reale n.1



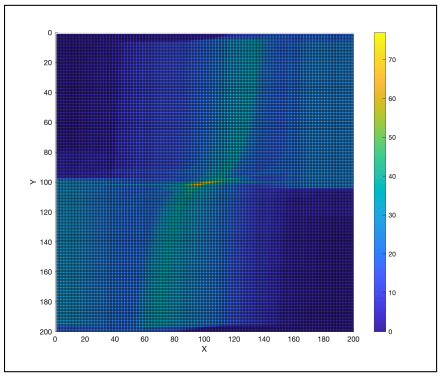
Input



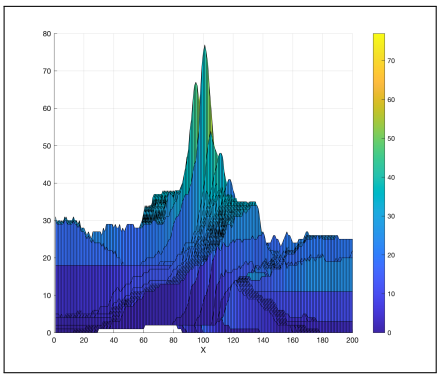
Input modificato



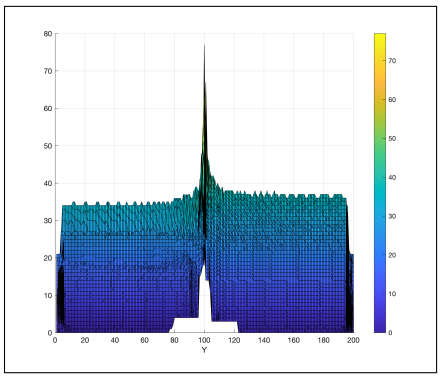
Isometrica



Sotto

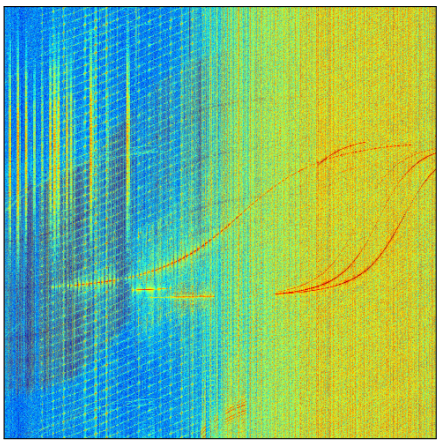


Vista da X

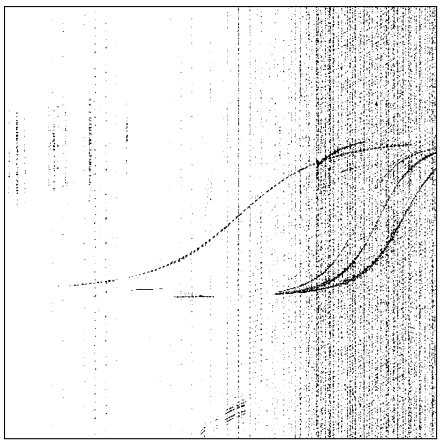


Vista da Y

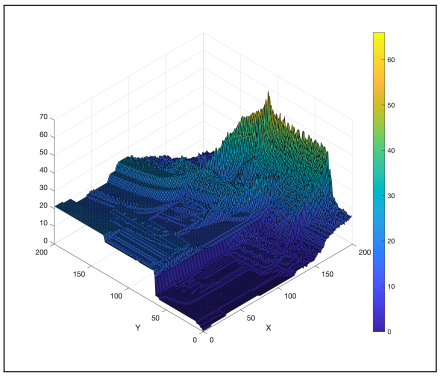
5.2.2 Esempio con immagine reale n.2



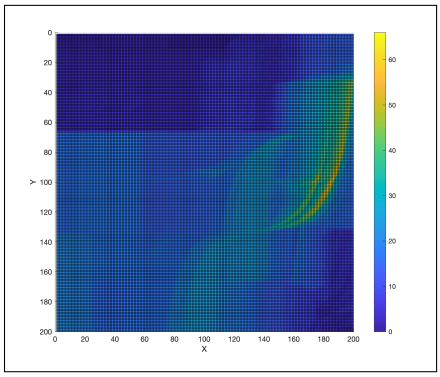
Input



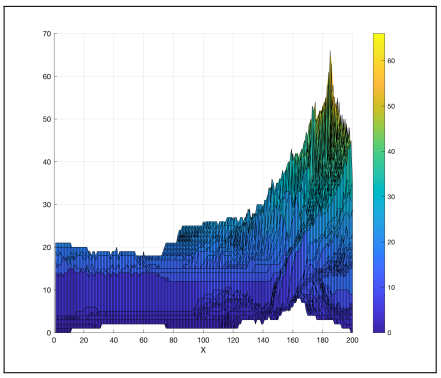
Input modificato



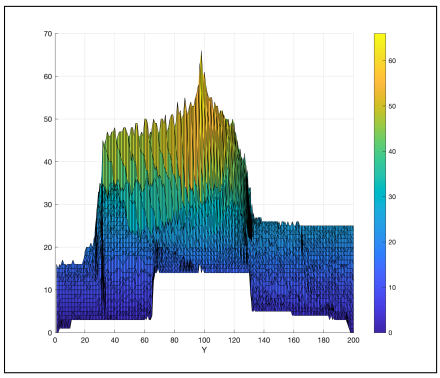
Isometrica



Sotto



Vista da X



Vista da Y

6 Conclusioni

Nei capitoli precedenti, attraverso gli esempi osservati, abbiamo approcciato lo stesso problema in diversi modi, utilizzando più di un metodo per migliorare la fruizione all'utente e lo sforzo mnemonico del sistema.

6.1 Approccio classico

Applicare direttamente tutti i principi dell'algoritmo di Hough, abbiamo visto essere inefficace agli obiettivi che ci siamo posti. Questo perché l'algoritmo originale prevede di avere una dimensione per ogni parametro della curva nell'immagine; ciò porta ad utilizzare un tensore quadridimensionale.

La rappresentazione di cinque dimensioni in modo comprensibile all'utente, cioè che non induca in errori di interpretazione, è pressoché impossibile.

Oltre questo, mantenere in memoria una quantità di dati abnorme e, su questi, ricercare massimi locali all'interno del tensore è anche computazionalmente complesso.

È di facile comprensione la grande quantità di valori presenti, nel caso si intenda analizzare un'immagine Full-HD[3] per un discreto numero di curve. Se ipotiziamo valori di a in un range $[0.1,\ 2]$ con passo di 0.05 e, d in un range [10, <numero di righe>] con passo 2, otteniamo facilmente:

$$\text{Numero di valori} = \underbrace{1920 \times 1080}_{\text{dim. immagine}} \times \underbrace{\frac{2 - 0.1}{0.05}}_{\text{valori di a}} \times \underbrace{\frac{1080 - 10}{2}}_{\text{valori di d}} \approx 42 \times 10^9$$

6.2 Approccio classico limitato

Successivamente abbiamo provato ad arginare questi due macro problemi fissando una dimensione. Abbiamo deciso di fissare d, in quanto è possibile che l'ampiezza delle curve sia dovuta ad una costante fisica: nel caso di satelliti non geostazionari, potrebbe essere la velocità relativa del satellite rispetto alla terra.

Per quanto riguarda il numero di valori, questo approccio permette di limitarli relativamente.

Nell'esempio di prima, possiamo togliere l'ultimo rapporto, lasciando:

$$\text{Numero di valori} = \underbrace{1920 \times 1080}_{\text{dim. immagine}} \times \underbrace{\frac{2 - 0.10}{0.05}}_{\text{valori di a}} \times \underbrace{1}_{\text{valori di d}} \approx 78 \times 10^6$$

[3]Le immagini Full HD hanno una risoluzione pari a 1920x1080 pixel.

Questo metodo, a differenza degli altri due, riduce anche le iterazioni necessarie per il computo delle occorrenze.

Mentre per quanto riguarda la visualizzazione, abbiamo trovato un metodo che, oltre alle tre dimensioni classiche, usa i colori.

Tuttavia, questo metodo crea un grafico che non è esattamente di immediata lettura e questo può portare a notevoli errori. Per avere un'informativa completa è necessario traslare i vari piani di taglio per visualizzare tutte le possibili combinazioni, questo richiede tempo e conoscenza da parte dell'utente.

6.3 Approccio innovativo

Il nostro approccio coniuga la semplicità rappresentativa del risultato con una ridotta necessità di memoria.

Questo perché la struttura che accoglie i dati è una semplice matrice bidimensionale. Essa è rappresentante dell'immagine, quindi abbiamo come coordinate della matrice quelle dei pixel.

Come abbiamo visto nell'algoritmo, il sistema salva i dati inerenti alla curva testata e al numero di occorrenze che questa ha generato, solo nel caso in cui non ci sia stata una curva già testata con maggior numero di occorrenze.

Facendo questo, ci accorgiamo che il numero di dati salvati è leggermente minore del secondo caso e sensibilmente del primo.

$$\text{Numero di valori} = \underbrace{1920 \times 1080}_{\text{dim. immagine}} \times \underbrace{3}_{\text{num. parametri}} \approx 6 \times 10^6$$

6.3.1 Fragilità principale

Come abbiamo visto nel sesto esempio delle immagini sintetiche e nel secondo esempio delle immagini reali, il nostro algoritmo, in caso di più curve ad S presenti nell'immagine, rischia di segnalare la presenza di curve in realtà inesistenti. Questi falsi positivi, sono dati dalla sovrapposizione di una curva più grande immaginaria con due più piccole davvero presenti.

Questo problema è intrinseco alla nostra scelta di rappresentare in un unico grafico i valori di tutte le curve a prescindere dai parametri delle stesse.

Riferimenti bibliografici

[1] Richard O. Duda and Peter E. Hart. Use of the Hough transformation to detect lines and curves in pictures. Commun. ACM, 15(1):11–15, January 1972. ISSN 0001-0782. doi: 10.1145/361237.361242. URL https://doi.org/10.1145/361237.361242. Ultimo accesso: 12 aprile 2024.

[2] Wikipedia. Tensore. URL https://it.wikipedia.org/wiki/Tensore. Ultimo accesso: 15 aprile 2024.

[3] D.H. Ballard. Generalizing the hough transform to detect arbitrary shapes. Pattern Recognition, 13(2):111–122, 1981. doi: 10.1016/0031-3203(81)90009-1. URL https://doi.org/10.1016/0031-3203(81)90009-1.

[4] Esempio di visualizzazione 4d. URL https://it.mathworks.com/help/examples/matlab/win64/Visualizing4DExample_04.png. Ultimo accesso: 11 aprile 2024.