APPLICAZIONE DELLA TRASFORMATA DI HOUGH ALL'ANALISI DEI SEGNALI DI ONDE GRAVITAZIONALI

Tesi di Laurea Triennale

Corso di Laurea in Fisica

Università degli Studi di Trieste

Godeas Leonardo
SM2000985

Relatore
Edoardo Milotti

Indice

Abstract 3

1 Onde gravitazionali 4

1.1 Introduzione teorica **4**

1.2 Sorgenti di onde gravitazionali **6**

1.3 L'interferometro **6**

1.4 Eventi d'interesse **9**

2 La trasformata di Hough 12

2.1 Introduzione teorica alla trasformata di Hough **12**

2.2 Scrittura del programma . . . . . **.15**

2.3 Utilizzo del programma su immagini di prova . . . . . **.23**

2.4 Utilizzo del programma sul segnale . . . . . **.26**

3 Valutazione dei risultati intermedi 27

3.1 La funzione di verosimiglianza **28**

3.2 Il criterio di informazione Bayesiano . . . . . **.29**

3.2 Scrittura della forma quadratica . . . . . **.30**

4 Risultati 32

Bibliografia 34

Abstract

L'osservazione diretta delle onde gravitazionali ha permesso di acquisire un nuovo punto di vista nello studio dell'Universo [1]: essa riguarda non solo la possibilità di registrare fenomeni inosservati precedentemente, come ad esempio la fusione di due buchi neri, ma costituisce anche una ulteriore fonte di informazioni per quanto riguarda fenomeni osservabili già osservabili in altro modo, ad esempio rivelandone la controparte elettromagnetica, permettendone dunque uno studio più approfondito. In questo lavoro verrà analizzato più nel dettaglio un evento di questo tipo, ossia la fusione di due stelle di neutroni rivelata il 17 agosto 2017, particolarmente rilevante poiché primo evento di cui si siano registrati sia la radiazione elettromagnetica che il segnale di onda gravitazionale [2].

Lo scopo di questa tesi è quello di mostrare come sia possibile, utilizzando un algoritmo basato sulla trasformata di Hough, interpolare l'evoluzione del segnale di un'onda gravitazionale nel piano tempo-frequenza partendo dai dati sperimentali forniti dagli interferometri.

Il lavoro è strutturato come segue: si introduce prima il significato di onda gravitazionale per poi spiegare brevemente il funzionamento della trasformata di Hough; in seguito vi è la descrizione del programma (scritto in Python) implementato per applicare la trasformata di Hough all'immagine associata allo spettrogramma dei dati sperimentali: vengono discusse le problematiche incontrate e le scelte compiute per migliorarne il funzionamento. In conclusione si discutono i risultati ottenuti ed i limiti del metodo qui descritto.

1. Le onde gravitazionali

1.1 Introduzione teorica

Le onde gravitazionali sono perturbazioni della struttura dello spaziotempo [3], che hanno natura ondulatoria e si propagano con velocità pari a quella dalla luce. La loro esistenza fu prevista già da Einstein nel 1916 sulla base della teoria della Relatività Generale, da lui stesso formulata negli anni precedenti. Questa teoria, assieme a quanto lo stesso Einstein aveva già formulato con la Relatività Speciale, ha modificato radicalmente il nostro modo di vedere l'universo: se prima spazio e tempo venivano visti come due realtà separate ed assolute, nella teoria di Einstein (e grazie anche all'interpretazione geometrica della Relatività Speciale di Minkowski) questi vengono unificati nell'unico concetto geometrico di spaziotempo e non sono più percepiti come assoluti, ma possono essere oggetto di deformazioni dovute alla presenza di una distribuzione di energia-momento.
Alla luce di questa interpretazione si può ottenere come masse in accelerazione una rispetto all'altra possano dare origine a fenomeni di tipo ondulatorio, detti per l'appunto onde gravitazionali, analogamente a come succede nel caso elettromagnetico, dove cariche accelerate generano onde elettromagnetiche.
A livello matematico il fenomeno è stato trattato attraverso una descrizione tensoriale[4][5]. In estrema sintesi, definendo $g_{\mu\nu}$ la metrica dello spaziotempo, ci si mette nelle condizioni per cui la metrica piatta euclidea $\eta_{\mu\nu}$

$$\eta_{\mu\nu} = \begin{pmatrix} -1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{pmatrix} \qquad (1.1)$$

subisce una perturbazione "piccola", ossia

$$g_{\mu\nu} = \eta_{\mu\nu} + h_{\mu\nu} \qquad (1.2)$$

dove il tensore simmetrico $h_{\mu\nu}$ è tale che $|h_{\mu\nu}| \ll 1$.

Trascurando i termini di ordine quadratico e superiore in *h* si cerca una soluzione nell'incognita $h_{\mu\nu}$ alle equazioni di campo di Einstein nel vuoto. Dopo un certo numero di conti si ottiene che $h_{\mu\nu}$ soddisfa l'equazione delle onde:

$$(\nabla^2 - \partial_t^2)h_{\mu\nu} = 0 \qquad (1.3)$$

dove $(\nabla^2 - \partial_t^2)$ è l'operatore D'Alembertiano.

Si può trovare che delle 16 componenti di $h_{\mu\nu}$ la Relatività Generale ne predice solo due indipendenti che vengono solitamente denotate come h_+ e $h_\times$: queste corrispondono alle due polarizzazioni associate alle onde gravitazionali.

La soluzione generale dell'equazione d'onda può essere descritta dunque come combinazione lineare delle due componenti h_+ e $h_\times$, introducendo due tensori di polarizzazione $\varepsilon_{\mu\nu}^{(1)}$ e $\varepsilon_{\mu\nu}^{(2)}$. Il tensore $h_{\mu\nu}$ diventa dunque:

$$h_{\mu\nu} = h_+\varepsilon_{\mu\nu}^{(1)} + h_\times\varepsilon_{\mu\nu}^{(2)} \qquad (1.4)$$

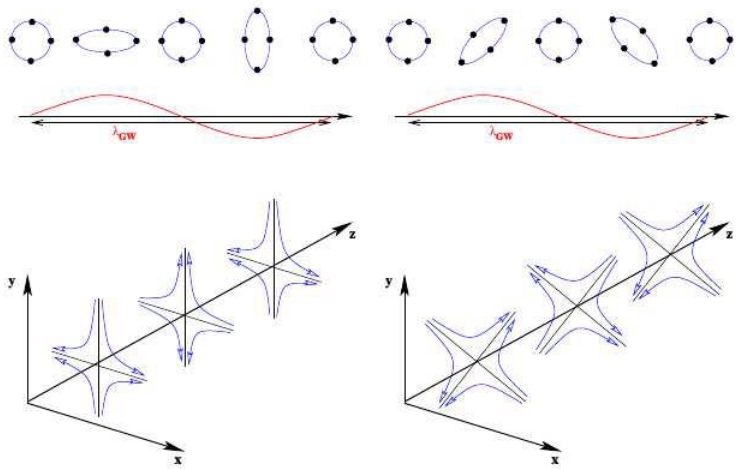

Figura 1: L'onda raffigurata si propaga lungo l'asse z ed agisce sul piano (x,y). A destra la componente + e a sinistra quella ×. La parte superiore mostra gli effetti delle due componenti su delle masse di prova disposte in cerchio.

1.2 Sorgenti di onde gravitazionali

Come detto in precedenza le onde gravitazionali sono associate a masse in accelerazione: tuttavia servono masse molto grandi per avere segnali rilevabili dagli interferometri, motivo per cui gli eventi finora registrati sono legati a particolari fenomeni astronomici.

Consideriamo ad esempio il caso di un sistema binario, che potrebbe essere dato da due buchi neri, due stelle di neutroni oppure un buco nero e una stella di neutroni (tutti fenomeni che sono stati attualmente rilevati [6]): i due corpi orbitanti percorrono una traiettoria a spirale, avvicinandosi progressivamente al centro di massa del sistema. Il processo di avvicinamento, che implica periodi di rotazione sempre più brevi, si conclude con la fusione dei due corpi, dopo la fusione la massa complessiva risulta essere inferiore alla somma delle masse iniziali dei due corpi: nel processo una parte della massa è stata rilasciata in forma di energia sotto forma di onda gravitazionale [7]. L'onda gravitazionale cresce in ampiezza e in frequenza man mano che ci si avvicina il momento della fusione: questo fa sì che nel piano tempo-frequenza l'onda gravitazionale da un fenomeno di questo tipo assuma una forma caratteristica, detta *chirp*.

Come detto in precedenza l'attuale sensitività degli interferometri consente di misurare solo fenomeni in cui viene rilasciata una sufficiente quantità di energia in onde gravitazionali: chiaramente, un ruolo importante viene giocato anche dalla distanza dalla Terra dei fenomeni rivelati.

Il fatto che un interferometro possa rivelare la deformazione dello spaziotempo causata dal passaggio di un'onda gravitazionale è un traguardo scientificamente notevole, raggiunto dopo vari anni di esperimenti e tentativi: se ne darà una breve spiegazione qui di seguito.

1.3 L'interferometro

Le misure che verranno riportate nel paragrafo successivo sono state raccolte presso gli interferometri americani LIGO (Laser Interferometer Gravitational-Wave

Observatory) di Livingston in Lousiana e di Hanford nello Stato di Washington, e presso Virgo, interferometro sito nel comune di Cascina, in provincia di Pisa.

Gli interferometri LIGO e Virgo sono, semplificando, una versione modificata dell'interferometro di Michelson. Quest'ultimo, progettato da Albert Abraham Michelson nel XIX secolo, è costituito da una sorgente luminosa, da un *beam splitter*, ovvero uno specchio semitrasparente, da un fotorivelatore e da due bracci perpendicolari tra loro al termine dei quali sono collocati due specchi completamente riflettenti [8]. Un fascio di fotoni *b* monocromatico viene generato dalla sorgente e prima di entrare nei due bracci attraversa il *beam splitter*, venendo in parte riflesso (b_R) e in parte trasmesso (b_T). L'inclinazione del *beam splitter* è tale che il fascio riflesso e quello trasmesso siano perpendicolari tra loro e ciascuno di essi percorra uno dei due bracci. Entrambi vengono rilessi dallo specchio collocato al termine del braccio e colpiscono nuovamente il *beam splitter*. A questo punto risulta necessario approfondire la struttura dello specchio semitrasparente: si tratta di un dispositivo ottico costituito da una lastra, che può essere di plastica o di vetro, in cui a una delle due facce è stato applicato un sottile strato di rivestimento; esso è costruito in modo che le quantità di luce trasmessa e riflessa, nel caso della configurazione illustrata, siano uguali. Se il fascio di luce colpisce il dispositivo dal lato del rivestimento la parte riflessa subisce uno sfasamento di 180 gradi, se invece la luce colpisce l'altro lato, ovvero la lastra, la parte riflessa non subisce sfasamento. Raggiungendo nuovamente lo specchio semitrasparente ciascuno dei due fasci di luce viene in parte riflesso in parte trasmesso, la parte trasmessa di b_R e la parte riflessa di b_T colpiscono il fotorivelatore e, nel caso il cammino ottico sia lo stesso per entrambi i bracci, si verifica interferenza distruttiva. Si ottiene lo stesso risultato anche nel caso in cui la differenza di cammino ottico nei due bracci sia un multiplo della lunghezza d'onda della luce del fascio, mentre nel caso in cui la differenza non sia né nulla né pari a un multiplo della lunghezza d'onda il fotorivelatore misura un segnale non nullo.

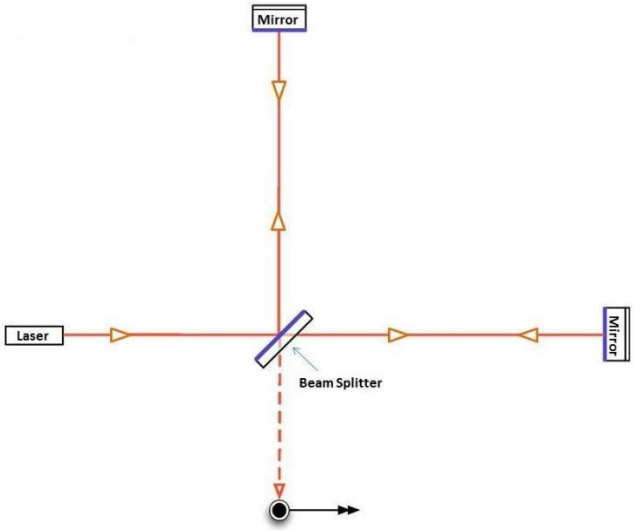

Figura 2: Rappresentazione semplificata di un interferometro di Michelson

Al passaggio di un'onda gravitazionale il cammino effettivo di ciascun fascio laser viene modificato per via della deformazione *h* dello spazio tempo causata dall'onda: questo è direttamente proporzionale alla differenza di fase rivelata alla ricombinazione dei due fasci che viene rivelata dal fotorivelatore.
Dunque l'output dell'interferometro può essere considerato come

$$h = \frac{\Delta L}{L} = \frac{\delta L_x - \delta L_y}{L} \qquad (1.5)$$

Come detto in precedenza, l'interferometro utilizzato per la misurazione delle onde gravitazionali presenta alcune modifiche rispetto a quello di Michelson: la principale è dovuta al fatto che la lunghezza efficace dei due bracci, le cui dimensioni sono già di per sé considerevoli (3 km per Virgo, 4 km per i due LIGO), viene aumentata di circa un fattore 100 tramite l'introduzione in ciascun braccio di una cavità di Fabry-Perot, ossia di due specchi paralleli con riflettività vicina a 1 tali per cui il fascio laser percorre un certo numero di volte ciascun braccio (dell'ordine del centinaio, per l'appunto) prima di ritornare al beam splitter[9]. Questo rende l'interferometro più sensibile al passaggio di un'onda gravitazionale, in particolare alle frequenze d'interesse per i sistemi binari sopra descritti [3].

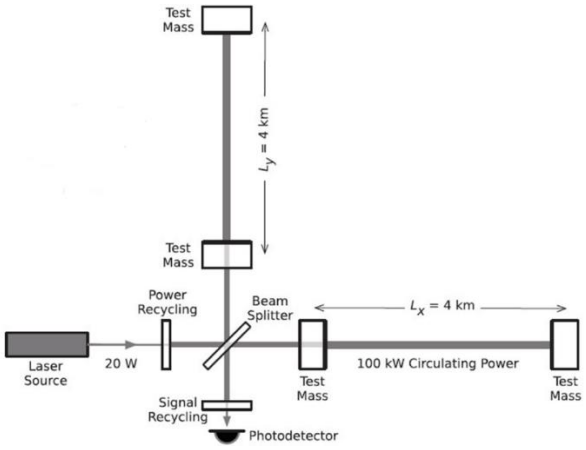

Figura 3: Rappresentazione semplificata di un interferometro per la rivelazione di onde gravitazionali

1.4 Eventi d'interesse

Il primo segnale di onda gravitazionale è stato registrato dagli interferometri LIGO di Livingston e di Hanford il giorno 14 settembre 2015, a quasi cento anni di distanza dalla formulazione da parte di Einstein dell'ipotesi dell'esistenza delle onde gravitazionali [1]. L'evento osservato è dato all'avvicinamento e alla fusione (*merger*) tra due buchi neri di masse stimate rispettivamente attorno alle 30 e 35 masse solari. La risposta dell'interferometro, che come detto risulta proporzionale alle deformazioni date dall'onda gravitazionale, è un segnale tipico di un sistema binario come quelli descritti in 1.2, la cui ampiezza e frequenza aumentano fino alla fusione dei due buchi neri per poi ridursi repentinamente. Per questo evento l'energia rilasciata dal sistema nel processo di fusione sotto forma di onde gravitazionali risulta essere equivalente a quella di tre masse solari.

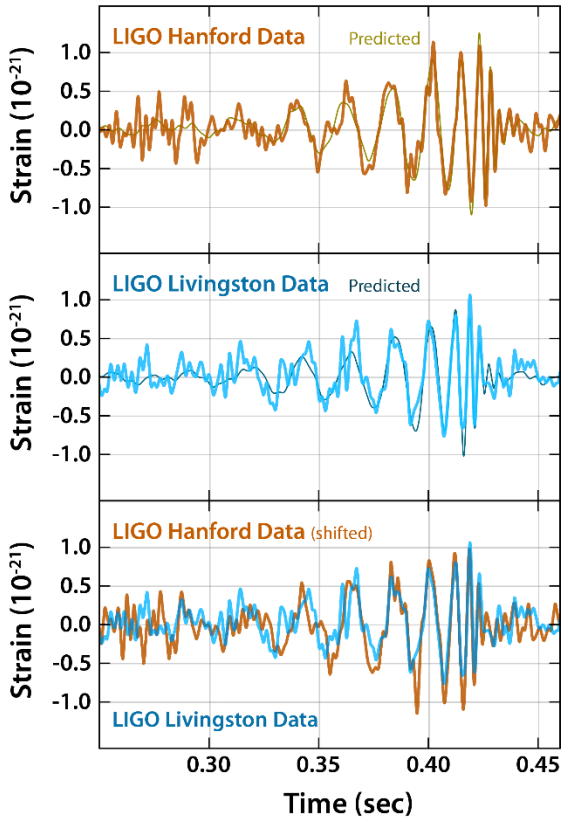

Figura 4: I tre grafici mettono in relazione tempo e deformazione. Il primo mostra il segnale misurato dal Ligo di Hanford con sovrapposto l'andamento atteso, il secondo mostra il segnale misurato dal Ligo di Livingston con sovrapposto l'andamento atteso, infine il terzo mostra la sovrapposizione tra i due, con i dati di Hanford traslati nel tempo.

Il segnale che verrà da qui in poi utilizzato è invece stato misurato il 17 agosto 2017[2]. In questo caso l'evento, registrato sia dagli interferometri LIGO che da Virgo, consiste nella fusione tra due stelle di neutroni: poiché in questo caso, a differenza del sistema di due buchi neri, viene emessa radiazione elettromagnetica, è stato possibile per la prima volta studiare lo stesso fenomeno sia tramite la sua parte gravitazionale sia tramite la controparte elettromagnetica, fornendo una quantità di informazioni senza precedenti sull'evento rivelato.

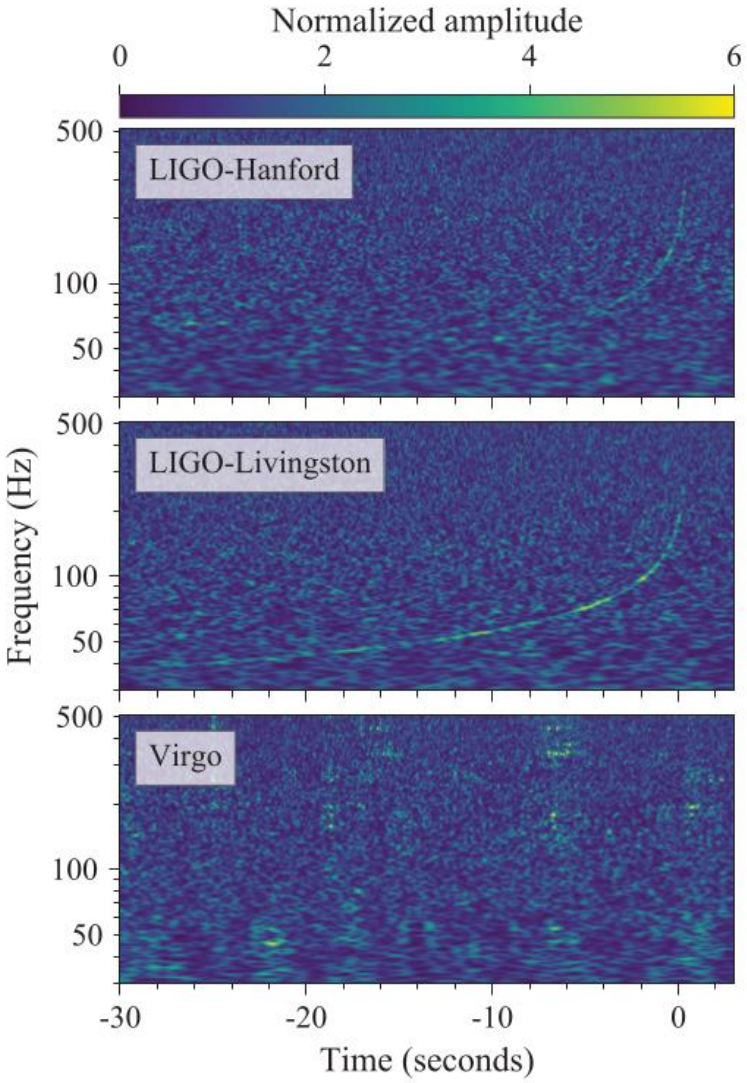

Figura 5: Spettrogrammi dei dati di LIGO Hanford, LIGO Livingston e Virgo di GW170817. Da notare come il rumore sia una parte rilevante dei dati, in particolare per l'interferometro Virgo, il quale ha una sensitività inferiore agli altri due.

Nell'analisi seguente i dati analizzati saranno quelli raccolti dall’interferometro LIGO Livingston riguardo a questo evento.

2. La trasformata di Hough

L'immagine che rappresenta la misura effettuata dall'interferometro è inevitabilmente affetta da rumore: lo scopo del programma trattato nei paragrafi successivi è quello di riuscire ad estrapolare dai dati una serie di punti nello spettrogramma che rappresentino l'andamento dell'onda gravitazionale, distinguendo questa dal sottofondo di rumore.
Per fare ciò si è scelto di utilizzare la trasformata di Hough, che verrà usata per implementare un algoritmo che sia in grado di riconoscere brevi segmenti che possano seguire la traccia dell'onda gravitazionale nel piano tempo frequenza.

2.1 Introduzione teorica alla trasformata di Hough

La trasformata di Hough è una procedura utilizzata nell'ambito dell'elaborazione digitale di immagini per individuare all'interno di queste forme o curve[10]. Nella sua formulazione più semplice permette di riconoscere linee rette, mentre nella sua forma generalizzata permette di riconoscere vari tipi di curve, tra cui circonferenze, ellissi e parabole. Da qui in poi verrà trattato l'utilizzo della trasformata di Hough per il riconoscimento di linee rette.
In uno spazio bidimensionale una retta può essere definita univocamente attraverso una coppia di valori *(θ, r)*, dove, posto un sistema di riferimento cartesiano, *r* è la distanza della retta dall'origine e *θ* è l'angolo che la perpendicolare alla retta forma con l'asse delle ascisse. In questo modo vale la relazione:

$$r = x\cos\theta + y\sin\theta \qquad (2.\ 1)$$

Al fine di garantire l'unicità della trasformazione si stabilisce che i domini siano:

$$x \in \mathbb{R}; y \in \mathbb{R}$$
$$r \in \mathbb{R}; \theta \in [0, \pi[\qquad (2.\ 2)$$

Possiamo dunque ricavare l'equazione della retta in forma esplicita:

$$y = -x\frac{\cos\theta}{\sin\theta} + \frac{r}{\sin\theta} \qquad (2.\ 3)$$

Lo spazio formato dai punti *(θ, r)* viene definito *Spazio di Hough*. A questo punto una retta *s* nel piano *(x,y)*, poiché è definita da (θ_s, r_s) in modo univoco, può essere posta in corrispondenza biunivoca con un punto nello Spazio di Hough attraverso la relazione:

$$\theta_s = \tan^{-1}\left(-\frac{1}{m_s}\right) \qquad (2.\ 4)$$

$$r_s = \frac{q_s}{\sqrt{m_s^2+1}} \qquad (2.\ 5)$$

dove abbiamo considerato l'equazione della retta s:

$$y = m_s x + q_s \qquad (2.\ 6)$$

Analogamente è possibile passare da un punto nello spazio di Hough ad una retta nello spazio *(x,y)*:

$$m_s = -\frac{\cos\theta_s}{\sin\theta_s} \qquad (2.\ 7)$$

$$q_s = \frac{r_s}{\sin\theta_s} \qquad (2.\ 8)$$

Considerando ora un fascio generico di rette si nota che, ad esempio, un fascio di rette parallele nello spazio *(x,y)* nello spazio di Hough viene definito dalla curva $\theta = \theta_0$, cioè da una retta parallela all'asse delle ordinate. Nel caso invece si

consideri un fascio di rette passanti per un punto $P = (x_p, y_p)$, esso sarà rappresentato, nello spazio di Hough, dalla curva:

$$r = x_p \cos\theta + y_p \sin\theta \qquad (2.9)$$

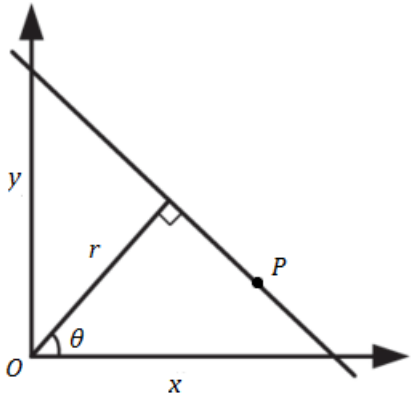

Figura 6: Una qualunque retta passante per il punto P, l'angolo θ e la distanza r corrispondono alle coordinate dello spazio di Hough

Proprio in questo consiste l'operazione di trasformazione, ovvero la trasformata di Hough opera facendo corrispondere ad una serie di punti dello spazio *(x,y)* una serie di curve sinusoidali nello spazio di Hough. Uno dei casi più semplici è quello in cui sono presenti solo due punti sul piano: in questo caso si hanno due curve nello spazio di Hough le quali hanno un punto di intersezione che a sua volta corrisponde ad una retta sul piano, che è proprio la retta passante per i due punti considerati in origine.

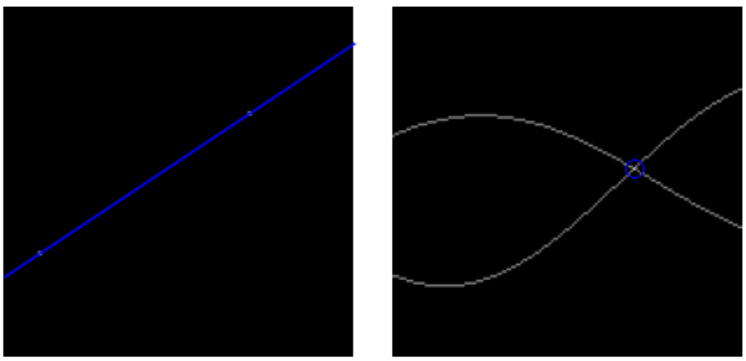

Figura 7: A sinistra lo spazio x-y, a destra lo spazio di Hough. In blu l'intersezione nello spazio di Hough che corrisponde a una retta nello spazio x-y.

Con tre punti non allineati le tre curve nello spazio di Hough hanno tre intersezioni che rappresentano tre rette, ciascuna delle quali passa per due dei tre punti.

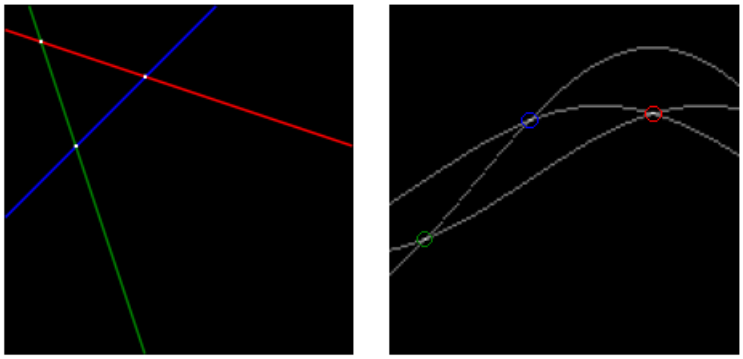

Figura 8: A sinistra lo spazio x-y, a destra lo spazio di Hough. Ciascuna delle tre intersezioni nello spazio di Hough indicate con colori diversi corrisponde alla retta del medesimo colore.

Nel caso si abbiano molti punti allineati le curve nello spazio di Hough hanno intersezione comune mentre nel caso di un insieme di punti di cui solo alcuni siano allineati solo alcune curve hanno un'intersezione comune, verranno interpolati in questo caso dalla retta corrispondente al massimo dell'array di accumulazione.

2.2 Scrittura del programma

L'applicazione di questa trasformata nell'ambito dell'elaborazione digitale di immagini consiste nella costruzione di un algoritmo che, a partire da un'immagine in scala di grigi (ovvero una matrice $n_0 \times n_1$ i cui valori sono compresi tra zero ed uno) in cui viene rappresentato un qualche tipo di linea, restituisca un insieme di segmenti che corrisponda il più possibile alle linee dell'immagine di partenza.
Per il momento consideriamo l'immagine non tanto come una matrice ma come una porzione rettangolare di piano contenente un certo numero di punti, le cui dimensioni dipendono da n_0 e n_1. Ora risulta necessario riscrivere i domini di *x* e *y*: nel caso generale espresso nel paragrafo precedente queste coordinate possono

avere come dominio tutto $\mathbb{R}$, mentre risulta evidente che nel caso qui considerato esse siano limitate dalle dimensioni dell'immagine e che dunque abbiano dominio finito nel caso in cui n_0 e n_1 siano finiti. Allo stesso modo risulta limitato il dominio di *r*, il dominio di θ invece rimane in ogni caso compreso tra *0* e π.
Si ha dunque che

$$x \in [-x_{max}, x_{max}]; y \in [-y_{max}, y_{max}]$$
$$r \in [-r_{max}, r_{max}]; \theta \in [0, \pi[\qquad (2.\ 10)$$

I domini di *x* e *y* così scritti indicano che l'origine degli assi è stata posta al centro dell'immagine: questa scelta non è l'unica possibile ma viene compiuta per minimizzare il valore di r_{max}.
Dovendo però operare su di una matrice è più conveniente ora fissare un nuovo sistema di assi, che denoteremo con *X* e *Y*. La *X* vale 0 sul lato sinistro del rettangolo e ha valore massimo sul lato destro, la *Y* vale zero sul lato superiore del rettangolo e ha valore massimo in quello inferiore.
Le formule per il cambio di coordinate quindi sono:

$$X = x + x_{max}, X \in [0, X_{max}] \quad con \quad X_{max} = 2x_{max}$$
$$Y = y_{max} - y, Y \in [0, Y_{max}] \quad con \quad Y_{max} = 2y_{max} \qquad (2.\ 11)$$

In questo caso righe e colonne della matrice corrispondono alle coordinate nello spazio *(X,Y)*, in modo analogo anche le due coordinate dello spazio di Hough possono corrispondere a righe e colonne di una matrice.
La dimensione di righe e colonne della matrice rappresentante lo spazio di Hough, che da ora in poi sarà chiamata *array di accumulazione*, può essere scelta arbitrariamente all'inizio del processo, a patto che venga corretta a tal proposito la funzione che converte le coordinate *(x,y)* in una serie di punti nello spazio di Hough.
La scelta della dimensione dell'array di accumulazione influenza il risultato ottenuto dal programma, dunque si esporranno in seguito le modalità utilizzate per stimare quale dimensione dell'array conduca al risultato più vicino a quello atteso.

Il codice qui proposto utilizza appunto una serie di funzioni che convertono coppie di coordinate dallo spazio *(x,y)* a quello di Hough e viceversa: due parametri stabiliscono rispettivamente il numero di righe e colonne dell'array di accumulazione; quest'ultimo viene generato "vuoto", composto cioè da soli zeri, ed ai suoi elementi sono assegnati dei valori secondo la procedura illustrata nel paragrafo seguente.

L'immagine viene inizialmente convertita in scala di grigi, cioè in una matrice *img0* (con 100 righe e 100 colonne) i cui valori sono compresi tra 0 e 1. Dopodiché vengono create le matrici *img1*, *img2* ed *img3* uguali tra loro e con lo stesso numero di righe e di colonne di *img0*, i cui valori possono essere solo 0 oppure 1: se l'elemento della matrice *img0* alla i-esima riga ed alla j-esima colonna è superiore al parametro *soglia_xy* si assegna al medesimo elemento delle matrici *img1*, *img2* ed *img3* il valore 0, altrimenti gli si assegna il valore 1.

Si definisce dunque l'array di accumulazione *acc* come un array composto da soli zeri con *ver* righe e *orr* colonne. Il valore del parametro *orr* viene scelto di volta in volta per ciascuna immagine, il valore del parametro *ver* dipende dal parametro *norm*, anche quest'ultimo scelto di volta in volta. Viene poi calcolato il parametro *diag*, prodotto di *norm* e del doppio del valore massimo che *r* può assumere (lunghezza della diagonale dell'array-immagine), *ver* è infine calcolato attraverso l'arrotondamento a numero intero di *diag*. Se si attribuisce il valore 1 al parametro *norm* il numero di righe dell'array di accumulazione corrisponderà alla lunghezza della diagonale dell'array-immagine, dunque i valori attribuiti a questo parametro saranno di poco superiori o inferiori all'unità.

2.2.1 La funzione *xy2h*

Si definisce la funzione *xy2h* come quella che riceve in input le coordinate di un punto nell'array-immagine e il valore *t* che va da 0 ad *orr*: questo valore, ossia l'indice di colonna dell'array di accumulazione, diviso per *orr* e moltiplicato per π rappresenta l'angolo nello spazio di Hough; in questo modo ci si assicura che l'argomento delle funzioni seno e coseno sia un valore compreso tra 0 e π. L'output della funzione è il valore di *r* nello spazio di Hough moltiplicato per il parametro *norm*. Il valore di *r* viene calcolato attraverso la prima formula, dove alla *x* corrisponde la colonna dell'array immagine e ad *y* corrisponde la differenza tra n_0

e la riga dell'array immagine. Alla luce dei cambi di coordinate prima illustrati e chiamando *i* la riga dell'array immagine e *j* la colonna, il calcolo di *r* viene svolto attraverso la seguente formula:

$$r = \left[\left(j - \frac{n1}{2}\right) \cdot \cos\left(\frac{\pi t}{orr}\right) + \left(\frac{n0}{2} - i\right) \cdot \sin\left(\frac{\pi t}{orr}\right)\right] \cdot norm \quad (2.12)$$

La riga dell'array di accumulazione viene calcolata attraverso l'arrotondamento di *diag* a numero intero e sottraendo ad esso l'output della funzione.

2.2.2 Le funzioni *h2xy* e *h2yx*

La funzione *h2xy* riceve in input i valori di riga e la colonna dell'array di accumulazione e la colonna dell'array immagine. Questa funzione effettua l'operazione inversa rispetto alla funzione precedente, ovvero partendo da un punto nello spazio di Hough e da un valore dell'ascissa dell'immagine (colonna dell'array immagine) calcola il valore dell'ordinata, in questo caso viene utilizzata la formula (2.3). In questo modo è possibile, partendo da un punto nello spazio di Hough, far variare il valore dell'ascissa ed ottenere la corrispondente ordinata, in modo da costruire una linea. Risulta però evidente che nel caso si abbia una retta verticale questo procedimento non conduca ad alcun risultato e risulti inefficace anche nel caso di rette a pendenza molto alta. Si rende dunque necessario introdurre una seconda funzione, denominata *h2yx*, che riceve in input la colonna anziché la riga dell'array immagine. La prima delle due funzioni viene utilizzata nel caso in cui la retta formi con l'asse delle ascisse un angolo compreso tra $-\pi/4$ e $\pi/4$, la seconda vien utilizzata per gli altri casi. Questa condizione si converte in una condizione su θ: la prima funzione viene utilizzata solo per i θ compresi tra $\pi/4$ e $3\pi/4$, la seconda per θ compresi tra 0 e $\pi/4$ oppure tra $3\pi/4$ e π. Si ottiene infine una condizione sull'indice colonna dell'array di accumulazione, che in questo caso viene chiamato *t*:

$$^1/_4 \leq t < {}^3/_4 \quad per \quad h2xy$$

$$0 \leq t < {}^1/_4 \quad oppure \quad {}^3/_4 \leq t < 1 \quad per \quad h2yx \qquad (2.\ 13)$$

2.2.3 La funzione *intorno*

Si definisce inoltre la funzione *intorno* con lo scopo di verificare la presenza di punti entro una certa distanza da un punto dato. Questa funzione riceve in input riga e colonna dell'array immagine (*x* e *y*) e restituisce la somma dei valori degli elementi dell'array *img1* contenuti in un quadrato di lato *(2n+1)* centrato nell'elemento dato, il valore di *n* è scelto arbitrariamente uguale a 2. Se l'elemento dato si trova in prossimità del bordo dell'array il quadrato di lato *(2n+1)* viene troncato, se si trova esattamente sul bordo (prima o ultima riga o colonna) l'output è zero. Il troncamento viene effettuato attraverso l'utilizzo della struttura *if, elif* ed *else* : se le coordinate dei punti sono tali che il quadrato di lato *(2n+1)* è contenuto interamente nell'array immagine allora per ognuna delle due coordinate dell'array vengono generati due valori, *x_sup* (oppure *y_sup*) uguale a *n*, *x_inf* (oppure *y_inf*) uguale a *1- n*, mentre se la differenza tra il numero di colonne e il valore della colonna ricevuto in input è inferiore ad *n* il valore *x_sup* viene posto uguale a questa differenza, allo stesso modo viene calcolato il valore *y_sup* nel caso in cui la differenza tra il numero di righe e il valore di riga in input sia inferiore ad *n*. Se invece il valore della colonna in input è inferiore ad *n* allora *x_inf* viene posto pari alla differenza tra colonna ed *n*, allo stesso modo è calcolato *y_inf* nel caso della riga. L'output della funzione è calcolato attraverso due cicli *for* facendo variare le colonne tra *x*+*x_inf* e *x*+*x_sup* e le righe tra *y*+*y_inf* e *y*+*y_sup*, sommando tra loro i corrispondenti valori dell'array immagine.

2.2.4 La funzione *riempi_intorno*

La funzione *riempi_intorno* svolge gli stessi passaggi della funzione *intorno* per quanto riguarda la generazione dei valori *x_inf*, *x_sup*, *y_inf* e *y_sup*, ma anziché sommare i valori che l'array assume all'interno del quadrato crea un array in cui

tutti gli elementi valgono uno e delle stesse dimensioni dell'array immagine; in seguito attribuisce dunque il valore zero agli elementi del quadrato di lato *(2n+1)* attraverso l'utilizzo di due cicli *for*.

2.2.5 La funzione *counting*

La funzione *counting* riceve in input un array qualunque e restituisce la somma di tutti i valori dell'array. Questa somma viene calcolata attraverso due cicli *for*, il primo dei quali va da zero al numero di righe dell'array, il secondo dei quali va da zero al numero di colonne dell'array.

2.2.6 La costruzione dell'array di accumulazione

Dopo aver definito queste funzioni si genera un array di accumulazione *acc1*, delle stesse dimensioni di *acc*. Esso viene riempito utilizzando solo alcuni degli elementi dell'array: viene generata una coppia di coordinate *(i,j)*, la prima viene fatta variare tra *0* ed *n0*, la seconda tra *0* ed *n1*, entrambe attraverso due cicli *for*. Se il valore dell'array *img1* a queste coordinate è pari ad 1 questa coppia di valori viene inserita come argomento della funzione *xy2h* unitamente ad una variabile *t* che viene fatta variare tra *0* e *orr* attraverso un ciclo *for*, ottenendo come risultato *r*. Si ottengono una serie di coppie di valori *(t,s)*, ciascuna delle quali corrisponde a una precisa coppia colonna-riga dell'array di accumulazione, all'elemento di queste coordinate, qualunque valore esso abbia, viene sommato il valore di *img0* alle stesse coordinate. Questo procedimento, per ottimizzare il programma, viene svolto contemporaneamente alla costruzione di *img1*, *img2* ed *img3*. All'interno dei due cicli *for*, si utilizza la condizione *if* per verificare, punto per punto, se il valore dell'array *img0* è superiore a *soglia_xy*. Le coordinate per le quali i valori di *img0* sono superiori a *soglia_xy*, sono le coordinate degli elementi degli array *img1*, *img2* ed *img3* (inizialmente generati vuoti ma delle stesse dimensioni di *img0*) ai quali viene assegnato il valore 1.
Al termine del processo il valore del parametro *soglia_h* viene ricalcolato moltiplicandolo per il massimo dell'array *acc1*.

2.2.6 L’individuazione dei segmenti sull’immagine

Inizialmente si definiscono quattro array: due vettori di dimensione 100 denominati *rec_s* e *rec_t* e due matrici di dimensione 100×2 denominate *estr_x* ed *estr_y*, un valore *i* pari a zero e una lista ordinata contenente i numeri interi tra *0* e *orr*.
Le operazioni seguenti sono reiterate al fine di individuare più linee, per interrompere la reiterazione si stabiliscono due condizioni: la prima condizione è posta sull’array *img2* e stabilisce di interrompere il ciclo qualora l’array rimanga completamente vuoto, composto cioè da soli zeri; la seconda condizione è contenuta nello stesso comando che genera il reiterarsi del processo ovvero la struttura *while*, si stabilisce cioè che il ciclo si interrompa all’individuazione di 100 linee, un numero molto maggiore di quello atteso nei casi presi in analisi: dunque questa condizione rappresenta un blocco di emergenza nel caso la prima condizione non funzioni, in particolare il vincolo sul numero di linee viene posto attraverso la condizione *i<100*, aggiungendo un’unità al valore *i* all’individuazione di ogni linea.
Vengono generati casualmente due valori interi *pix0* e *pix1*, il primo compreso tra *0* ed *n0* ed il secondo compreso tra *0* ed *n1*, si definisce *pix* il valore di *img2* a queste coordinate. La scelta di generare un nuovo valore è resa necessaria dal fatto che subito dopo l’elemento di *img2* alle stesse coordinate viene posto uguale a zero. Nel caso in cui questo valore sia pari ad 1 si producono le coppie *(t,s)* secondo il procedimento già illustrato, ma i valori di *t* non vengono fatti variare in modo ordinato, bensì viene modificato in modo casuale l’ordine degli elementi della lista *l* attraverso l’istruzione *shuffle* appartenente al pacchetto *random*. La scelta di far variare *t* tra i valori di *l* attraverso un ciclo *for* piuttosto di farlo semplicemente variare tra 0 e *orr* impedisce che, qualora ci fossero più valori vicini tra loro superiori alla soglia stabilita, non venga selezionato per primo sempre l’elemento dell’array di accumulazione con il più basso indice di colonna.
Quando viene individuato un elemento dell'array di accumulazione che supera *soglia_h* si cerca una linea nell'immagine, il primo passaggio è la creazione di un array *canc* delle stesse dimensioni dell’array immagine ma composto di soli uni, è necessario svolgere questo passaggio in questo momento in modo che, con la reiterazione del processo, per ogni nuovo elemento dell’array di accumulazione preso in considerazione tutte le modifiche fatte all’array *canc* vengono eliminate.

Nel caso in cui t sia compreso tra *orr/4* e *3orr/4* (ovvero θ compreso tra $\pi/4$ e $3\pi/4$) si generano due valori *xc* e *xd* i quali corrispondono a *pix1* rispettivamente aumentato e diminuito di un'unità, a partire da questi si generano rispettivamente *yc* e *yd* utilizzando la funzione *h2xy* con argomento le coppie *(t,s)* ed uno dei due valori generati a partire da *pix1*, poi si generano le copie di questi nuovi valori: *xc_*, *xd_*, *yc_* e *yd_*. A questo punto viene introdotto un ciclo *while* che ha come condizione che la funzione intorno abbia valore maggiore di zero per gli elementi dell'array immagine alle coordinate di *xd* e *yd*, l'array *canc* viene moltiplicato per il valore della funzione *riempi_intorno* calcolato per le suddette coordinate. La moltiplicazione fra array in *python* non è un prodotto fra matrici e pertanto è possibile anche tra matrici non quadrate a patto che le due abbiano stesse dimensioni, essa restituisce un array i cui elementi sono il prodotto dei due elementi degli array di partenza alle stesse coordinate.

Il risultato della reiterazione di tale operazione è un array *canc* che ha "memorizzato" le coordinate in cui la funzione *riempi_intorno* ha assunto il valore zero in ogni ciclo, ciò significa che al termine della reiterazione vari elementi dell'array *canc* hanno assunto il valore zero, in corrispondenza degli elementi dell'array che sono stati coinvolti nel processo.

Il ciclo prosegue con la ridefinizione delle copie di *xd* e *yd*, cioè *xd_* e *yd_* vengono nuovamente eguagliati ai loro omologhi, al primo ciclo questa operazione risulta non necessaria poiché già svolta, ma assume importanza successivamente al ripetersi del processo ovvero al variare dei valori di *xd* e *yd*. Successivamente si procede con la sottrazione di un'unità a *xd* ed il successivo calcolo di *yd* sempre attraverso la funzione *h2xy*. In questo modo al ciclo successivo la funzione *intorno* viene calcolata in un diverso punto ed il processo si ripete fintanto che la funzione non restituisce il valore zero, ovvero finché si giunge in prossimità del bordo dell'array oppure finché tutti gli elementi appartenenti al quadrato di lato *(2n+1)* centrato nelle coordinate *xd* e *yd* hanno valore pari a zero.

Un analogo ciclo *while* viene compiuto successivamente, esso svolge le stesse operazioni di quello precedente salvo per il fatto che agisce su *xc* e *yc* e si aggiunge (anziché sottrarre) un'unità a *xc*.

Al termine di tutte queste operazioni si ottengono quattro valori di *xc_*, *xd_*, *yc_* e *yd_* corrispondenti agli estremi di un segmento. Il motivo per cui non si assumono *xc*, *xd*, *yc* e *yd* come estremi è che al termine del processo le loro coordinate

conducono ad un segmento più largo del necessario poiché il ciclo *while* si interrompe per coordinate che non soddisfano la condizione posta. Allo stesso modo l'eguaglianza tra *xc_*, *xd_*, *yc_*, *yd_* e *xc*, *xd*, *yc*, *yd* è posta anche prima dei cicli while perché qualora non lo fosse e se una delle due coppie di coordinate non soddisfacesse le condizioni allora alcuni dei valori tra *xc_*, *xd_*, *yc_* e *yd_* non sarebbero mai definiti.
Finora è stato esaminato il caso in cui *t* è compreso tra *orr/4* e *3orr/4*, nel caso in cui *t* assuma altri valori si procede in modo del tutto analogo, con la differenza che i valori ad essere modificati sono *yc* e *yd* e gli altri due sono calcolati attraverso la funzione *h2yx*.
Concluso il processo si calcola la lunghezza del segmento, definita come:

$$lun = \sqrt{(xc\_ - xd\_)^2 + (yc\_ - yd\_)^2} \qquad (2.\ 14)$$

Se il valore di *lun* supera il valore del parametro *soglia_linea*, stabilito arbitrariamente, allora si registrano tutti i valori ottenuti: i due elementi presenti su ogni riga di *estr_x* vengono eguagliati a *xc_* e *xd_*, allo stesso modo si costruisce *estr_y*, mentre *s* e *t* si registrano nell'array *rec_s* e *rec_t*.

2.3 Utilizzo del programma su immagini di prova

Per valutare l'efficacia dell'algoritmo si prendono inizialmente in considerazione una serie di casi molto semplici. Per questo motivo sono state scattate 4 immagini di prova nelle quali è presente un solo segmento nero su sfondo bianco.
I segmenti sono stati scelti con inclinazioni diverse, alcuni con uno o entrambi gli estremi sul bordo dell'immagine o con un estremo all'origine degli assi in modo da descrivere più casi diversi possibile. Le immagini sono state ridotte ad una risoluzione di 100 pixel per lato per velocizzare il funzionamento del programma ed i colori sono stati invertiti, cioè si è preso il loro negativo, per migliorare l'impatto visivo e non dover cambiare la scrittura del programma una volta passati all'analisi del segnale vero e proprio. Per quanto riguarda questa fase di prova la valutazione dei risultati consiste nel ricercare i valori per i quali si ottenga una sola

linea sovrapposta per la sua interezza ai punti dell'immagine, nelle fasi successive si descriverà un più oggettivo criterio per la valutazione dei risultati ottenuti.

2.3.1 Scelta del valore soglia_xy

Viene scelto in questo caso il valore 0.5 in quanto le immagini utilizzate filtrate a questo valore mostrano molto chiaramente la linea che si vuole individuare e poco disturbo attorno ad essa. Questo valore va bene per tutte le quattro immagini di prova, ma è necessario verificare per ogni nuova immagine utilizzata il valore ottimale di questo parametro.

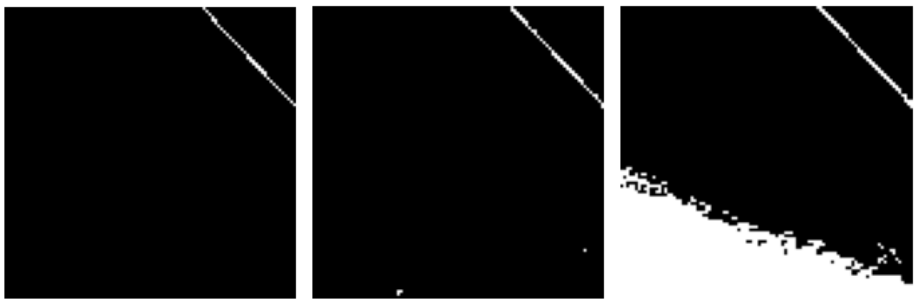

Figura 8.: le figure mostrano l'immagine 3 rispettivamente con 0.4, 0.5 e 0.6 valori di *soglia_xy*

2.3.2 Scelta del valore soglia_linea

Il valore *soglia_linea* viene posto uguale a 10 per tutte le immagini. Nei casi in cui le linee ottenute siano più di una ma solo una di queste segua il segmento dell'immagine per tutta la sua lunghezza è possibile aumentare il valore di *soglia_linea* per escludere le altre.

2.3.3 Scelta dei valori orr e norm

Questi parametri determinano la dimensione dell'array di accumulazione. Se troppo bassi i valori di *r* e *t* individuati dall'algoritmo si discosteranno sensibilmente da quelli reali, cioè il segmento tracciato non combacerà molto bene con quello dell'immagine, se troppo alti comprometteranno la giusta individuazione dei massimi nell'array di accumulazione. Per questi parametri non viene indicato un unico valore, ma infatti essi verranno modificati a seconda dell'immagine. La scelta

in questi primi test su immagini di prova viene compiuta in modo arbitrario: in seguito verrà esposto il criterio utilizzato nel caso del l'analisi dell'immagine contenente il segnale.

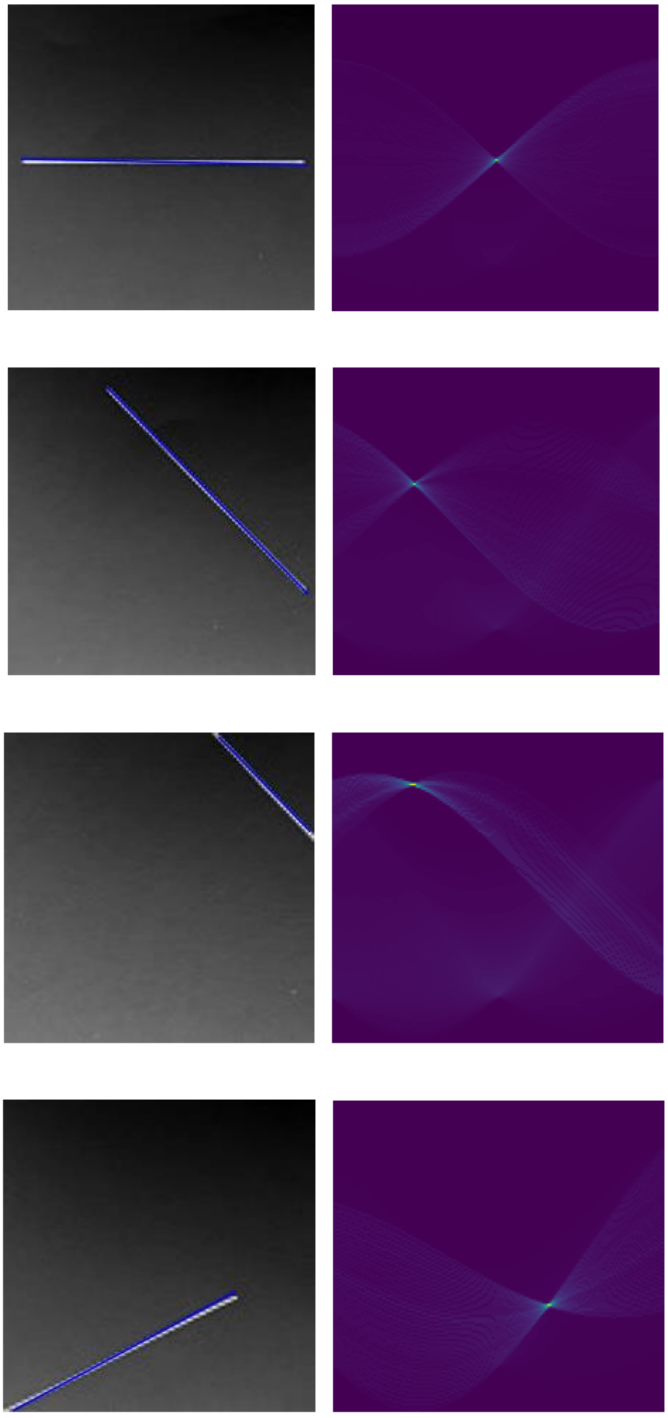

Figura 10: A sinistra lo spazio x-y, a destra lo spazio di Hough. Ai punti nello spazio x-y sono sovrapposte le linee trovate, lo spazio di Hough viene raffigurato in viola e verde anziché nero e bianco per un maggiore contrasto.

Come esempio nella figura 10 viene mostrato per ogni immagine il rispettivo spazio di Hough e la linea ottenuta dal programma viene sovrapposta all'immagine originale. I valori di *orr* e *norm* sono stati stabiliti in modo arbitrario e sono rispettivamente 150 e 1.

2.4 Utilizzo del programma sul segnale

A questo punto si ha un programma correttamente funzionante, il passo successivo consiste nell'applicarlo al segnale. L'immagine rappresentante il segnale si discosta però da quelle di prova per diverse caratteristiche: la sua risoluzione è molto maggiore, motivo per cui si è scelto di lavorare su una versione dell'immagine a risoluzione ridotta; i valori dell'array sono tali che sia in questo caso preferibile assegnare al parametro *soglia_xy* il valore di 0.4 anziché 0.5; la curva che ci si prefigge di individuare è appunto una curva e non un segmento di retta, verrà dunque rappresentata da una serie di segmenti e i massimi da individuare nello spazio di Hough saranno più di uno.

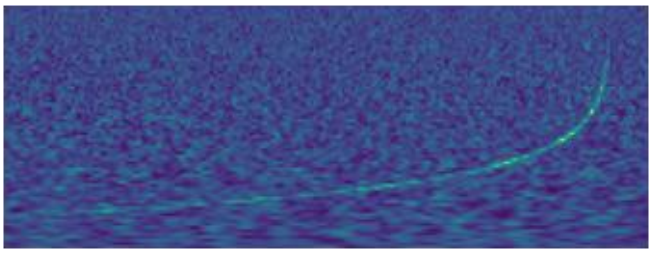

Figura 11: La versione a risoluzione ridotta dell'immagine.

3. Valutazione dei risultati intermedi

Risulta evidente che il risultato ottenuto vari a seconda del valore di alcuni parametri stabiliti arbitrariamente. I parametri *soglia_xy* e *soglia_linea* sono stati scelti in base al funzionamento del programma su immagini di prova, il valore di *soglia_linea* viene mantenuto pari a 10, mentre *soglia_xy* è stato modificato nelle modalità esposte nella sezione 2.4. In entrambi i casi il valore dei parametri non sarà modificato da ora in poi. Si valuterà invece il funzionamento del programma per diverse combinazioni di valori dei parametri *orr* e *norm*, che determinano le dimensioni dell'array di accumulazione e di conseguenza influenzano l'output del programma.

Il risultato del processo è un set di segmenti sovrapposti all'immagine di partenza: ad ogni set corrisponderà una certa coppia di valori dei parametri *orr* e *norm*, ciò che si cerca di ottenere è un set che segua il più possibile l'andamento dei punti presenti in *img1* e al contempo individuare la corrispondente coppia di valori dei parametri.

Una prima modalità di verifica può essere il calcolo di una forma quadratica, che tenga conto degli scarti tra valore della linea e punto sull'immagine. Ogni segmento viene percorso in tutta la sua lunghezza e per ogni pixel si calcola la distanza con il punto dell'immagine *img1* più vicino. A ciascun pixel di ciascun segmento corrisponderà una distanza, la forma quadratica sarà data dalla somma dei quadrati di tutte le distanze. Così si va a stabilire un criterio che quantifichi quanto i segmenti trovati si discostano dall'immagine originale.

Questo criterio risulta però inefficace in alcuni casi: può accadere infatti che per diverse combinazioni di parametri si ottenga un alto numero di linee le quali andrebbero sì a coprire la figura in analisi, ma in modo ridondante sovrapponendosi tra di loro. In questo caso la forma quadratica assumerebbe comunque un valore maggiore rispetto a diverse configurazioni a causa del sovrapporsi delle linee, pur non essendo una configurazione ottimale al fine di descrivere la figura in analisi. Si rende quindi necessaria l'introduzione di un ulteriore termine che vada a considerare anche il numero di linee trovate dal processo. Per fare ciò si considera una forma quadratica che possa dipendere anche da questi parametri facendo uso del *criterio di informazione Bayesiano* (*Bayesian information criteria*, BIC), il

quale somma alla funzione di log-verosimiglianza un termine dipendente dal numero di osservazioni e dal numero di parametri.

3.1 La funzione di verosimiglianza

In statistica la *funzione di verosimiglianza*, o di *likelihood*, è una funzione dei parametri dato un set di dati ed è collegata alla distribuzione di probabilità di questi dati. Dato un campione costituito da *n* misure di una stessa variabile *X* espresso sotto forma di vettore $\vec{X}$ e un insieme di parametri definiti dal vettore $\vec{\theta}$, la funzione di distribuzione della variabile *X* è definita in modo generico come $f\left(X, \vec{\theta}\right)$. La funzione di verosimiglianza è come detto una funzione dei parametri $\vec{\theta}$ dato il set di dati $\vec{X}$e può essere scritta come, assumendo indipendenza tra i vari X_i:

$$L\left(\vec{X}, \vec{\theta}\right) = \prod_{i=1}^{n} f\left(X_i, \vec{\theta}\right) \qquad (3.\ 1)$$

Questa funzione viene spesso utilizzata per ricavare gli stimatori, passando attraverso il principio di massima verosimiglianza (Maximum Likelihood): esso afferma che la più accurata stima dei parametri $\vec{\theta}$ è quella che massimizza la funzione di verosimiglianza. Per comodità è possibile utilizzare il logaritmo della likelihood: in questo caso la funzione viene detta log-verosimiglianza

$$logL\left(\vec{X}, \vec{\theta}\right) = \sum_{i=1}^{n} \ln\left(f\left(X_i, \vec{\theta}\right)\right) \qquad (3.\ 2)$$

Nel caso la variabile abbia funzione di distribuzione gaussiana, con σ_i deviazione standard di X_i e μ_i il valore di aspettazione di X_i, la funzione di verosimiglianza e la log-verosimiglianza prenderanno la forma:

$$L\left(\vec{X}, \vec{\theta}\right) = \prod_{i=1}^{n} \frac{1}{\sqrt{2\pi\sigma_i^2}} e^{-\frac{\left(X_i - \mu_i\right)^2}{2\sigma_i^2}} \qquad (3.\ 3)$$

$$logL(\vec{X},\vec{\theta}) = \sum_{i=1}^{n}\left[-\frac{1}{2}\ln(2\pi{\sigma_i}^2) - \frac{(X_i-\mu_i)^2}{2{\sigma_i}^2}\right] \quad (3.\ 4)$$

La suddetta likelihood gaussiana può essere applicata al nostro caso: infatti, la posizione stimata dei punti che compongono le linee segue, approssimativamente, una distribuzione gaussiana.

Per poter adattare queste funzioni al caso preso in analisi bisogna innanzitutto stabilire come selezionare i valori di X_i, μ_i e σ_i dal risultato ottenuto: per un certo valore di indice colonna μ_i può essere considerato come il valore di indice riga corrispondente individuato sul segmento restituito dal programma, mentre X_i si può considerare come il valore di indice riga del punto più vicino, cercato nella stessa colonna; oppure, in modo del tutto analogo, è possibile partire dall'indice riga e considerare i due valori come i valori di indice colonna. Il valore di σ_i è più difficile da considerare, ma è ragionevole porlo uguale a 1, anche in base ad alcune simulazioni preliminari svolte, e il termine $-\frac{1}{2}\ln(2\pi{\sigma_i}^2)$ all'interno della sommatoria viene trascurato. Nel nostro caso i parametri da cui dipendono le X_i sono *orr* e *norm*: la likelihood verrà dunque massimizzata rispetto a questi due, ossia ripetendo il procedimento per vari valori dei parametri e scegliendo la coppia che massimizza la lilelihood, infine massimizzare la funzione di likelihood equivale in questo caso a minimizzare la forma:

$$\sum_{i=1}^{n}\left[\frac{(X_i-\mu_i)^2}{2{\sigma_i}^2}\right] \quad (3.\ 5)$$

3.2 Il criterio di informazione Bayesiano

Derivato da Gideon Schwartz nel 1978, il *criterio di informazione Bayesiano* o *Bayesian information criteria* (BIC) ha la seguente formulazione [12]:

$$BIC = -2\log L_{max} + k\ln N \quad (3.\ 6)$$

dove *k* è il numero di parametri di un modello e *N* il numero di osservazioni. Esso viene utilizzato per la valutazione dell'efficacia di un modello, infatti la dipendenza dal numero di parametri introduce una selezione che permette di riconoscere i parametri che possono essere esclusi al fine di ottenere il modello più efficiente. Questo tipo di criterio viene appunto utilizzato nel caso in cui si volesse interpretare un set di dati e non si conoscesse il numero ottimale di parametri da utilizzare, inoltre è possibile valutare se sia o meno proficua l'introduzione di un nuovo parametro. Nel caso della forma quadratica discussa in seguito il numero di parametri corrisponde al numero di segmenti trovati.

3.3 Scrittura della forma quadratica

L'ultima parte del programma implementa una struttura per il calcolo di una forma quadratica. Anche in questo caso si utilizza un ciclo *for* che viene fatto andare da zero al numero di linee ottenute, per ogni linea la struttura *if* introduce una condizione sull'inclinazione della linea: se è maggiore di 45° si prosegue in un modo, se inferiore in un altro.

Per evitare ripetitività si esporrà ora il funzionamento del programma nel caso in cui l'inclinazione sia minore di 45°, ricordando che l'altro caso è identico a meno di un'inversione delle coordinate degli array.

Un ciclo *for* fa variare la coordinata *x0* tra i valori compresi tra gli estremi del segmento, la corrispondente coordinata *y* viene calcolata attraverso la funzione *h2xy* con valori di *r* e *t* ricavati dagli array *rec_s* e *rec_t*. A questo punto un ciclo *while* fa variare i valori di *y* in due modi: *y0_d* viene diminuito di un'unità ad ogni passo, *y0_c* aumentato, due condizioni *if* impediscono alle due variabili di assumere valori negativi o superiori a quelli ammessi dall'array-immagine. Il ciclo *while* viene interrotto quando viene ritrovato un valore diverso da zero sull'array *img31* in corrispondenza delle coordinate *x0,y0_c* oppure *x0,y0_d*.

L'array *img31* è stato definito all'inizio del processo come differenza tra *img3*, che è l'array "in bianco e nero" originale, ed *img1*, che è lo stesso array ma a cui sono stati poi rimossi i punti interessati nel processo di individuazione delle linee secondo quanto espresso nel capitolo precedente, perciò *img31* risulta contenere i

soli punti interessati nel processo di individuazione delle linee. La scelta di utilizzare img31 anziché img3 è dipesa dal fatto che l'individuazione di un punto su quest'ultimo array poteva essere attribuita non solo ai punti interessati nel processo di individuazione delle linee, ma anche al cosiddetto disturbo.

La variabile *count* è utilizzata per "contare" il numero di passi compiuti da un singolo ciclo *while* e questo numero viene elevato al quadrato al termine di ciascun ciclo e poi sommato a *chi*, una variabile posta uguale a zero all'inizio del processo. In questo modo, ripetendo il processo per ogni punto della suddetta linea e poi per ogni linea si ottiene una forma quadratica analoga alla funzione di log-verosimiglianza (moltiplicata per -1).

Infine utilizzando la funzione *counting* si calcola la differenza tra il numero di punti in *img3* e quello in *img1*.

La forma quadratica scelta è data dalla somma tra la forma quadratica sopracitata ed il numero di linee, registrato dalla variabile *i*, per privilegiare le soluzioni con un minor numero di linee, moltiplicato per due: in questo modo verrà privilegiato il set di parametri che minimizzi la forma quadratica.

4. Risultati

Dopo aver utilizzato il programma sul segnale facendo variare i valori di *orr* e *norm* si determina la coppia di valori per questi due parametri che minimizzi la forma quadratica, in particolare i valori scelti sono:

$$orr = 200$$
$$norm = 2$$

Da notare come la serie di segmenti segua abbastanza bene la parte iniziale della curva in cui la pendenza varia meno velocemente, mentre la parte finale in cui la pendenza varia più repentinamente non viene ben identificata dal programma. La parte iniziale è dunque di più facile individuazione poiché alcuni tratti della curva possono essere visti come quasi rettilinei, e dunque costituiranno i massimi dell'array di accumulazione. I segmenti nel tratto iniziale però, pur seguendo bene l'andamento della curva, non la coprono completamente, a causa del fatto che all'interno del segnale la curva è più riconoscibile nel suo tratto finale, poiché l'aumento di ampiezza dell'onda gravitazionale fa sì che i punti qui si discostino maggiormente dal rumore, mentre il tratto iniziale mostra zone non ben distinguibili dal rumore. Il risultato è che il tratto finale sia quasi interamente coperto da segmenti, ma che essi siano collocati in modo più impreciso rispetto agli altri, per il fatto che la pendenza della curva diventa sempre maggiore.

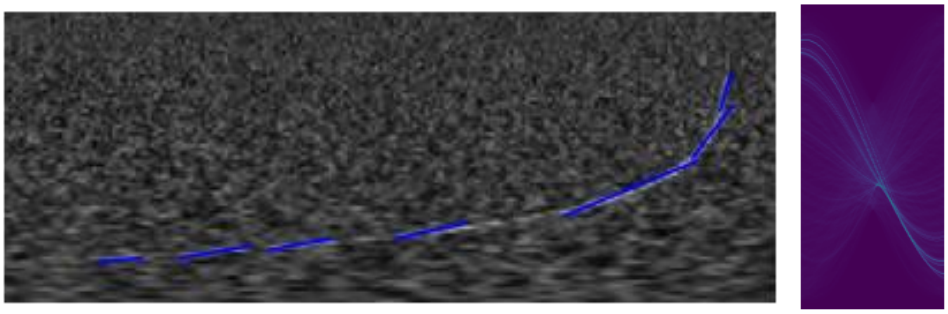

Figura 12: A sinistra lo spazio x-y, a destra lo spazio di Hough.
Ai punti nello spazio x-y sono sovrapposte le linee trovate.

Risulta d'interesse anche la raffigurazione dell'array *img31*, in cui sono presenti solo i punti interessati dal processo di selezione delle linee, in quanto in essa è riconoscibile la curva che definisce il segnale priva del precedente rumore.

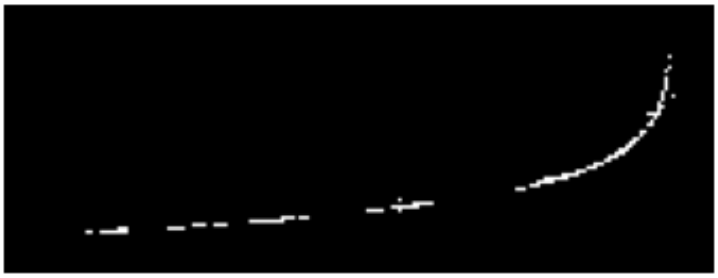

Figura 13: L'array *img31* contiene solo i punti interessati dal programma.

È dunque possibile affermare che un programma che applichi un algoritmo basato sulla trasformata di Hough ad un segnale di onda gravitazionale sia effettivamente in grado di selezionare la curva tempo-frequenza, ricercata a partire dallo spettrogramma, trasformata già utilizzata in altri studi per l'analisi della frequenza delle onde gravitazionali [11].

Tuttavia è necessario altresì evidenziare i limiti di questo programma: il tempo che esso impiega a giungere ad un risultato è di alcuni secondi ed aumenta considerevolmente con l'aumentare della risoluzione delle immagini utilizzate. Il metodo scelto per la valutazione dei risultati può essere applicato efficacemente nel caso in oggetto, si è scelto di sfruttarne il funzionamento facendo variare due parametri entro una griglia di valori, mentre il valore dei parametri *soglia_xy* e *soglia_linea* è stato fissato all'inizio. Applicando il programma ad immagini diverse da quelle utilizzate potrebbe rendersi necessario modificare i valori di questi ultimi e selezionarne i migliori, ma ciò significherebbe rendere il programma computazionalmente più dispendioso dovendo minimizzare la forma quadratica su quattro (e non due) parametri.

Il set di segmenti e l'array *img31* sono dunque i risultati che il programma restituisce. Essi possono essere utilizzati per realizzare un fit della curva tempo frequenza, il che consente di individuare tracce di onde gravitazionali all'interno dei dati degli interferometri. Per questo motivo lo studio dello spettrogramma riveste un particolare interesse ed in quest'ottica vanno intesi anche i risultati ottenuti dal programma.

Bibliografia

1. B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), *Observation of Gravitational Waves from a Binary Black Hole Merger*. Phys. Rev. Lett. 116, 061102 (2016)

2. B. P. Abbott et al. (LIGO Scientific Collaboration and Virgo Collaboration), *GW170817: Observation of Gravitational Waves from a Binary Neutron Star Inspiral.* Phys. Rev. Lett. 119, 161101 (2017)

3. Maggiore, Michele. *Gravitational waves: Volume 1: Theory and experiments*. OUP Oxford, 2007.

4. Eanna E. Flanagan, Scott A. Hughes, *The basics of gravitational wave theory*, New J.Phys. 7 (2005) 204

5. Philippoz, Lionel Antoine, *On the polarization of gravitational waves*, University of Zurich, Faculty of Science (2018)

6. Abbott, R., et al., *GWTC-3: Compact binary coalescences observed by LIGO and Virgo during the second part of the third observing run.*, arXiv preprint arXiv:2111.03606 (2021).

7. LIGO Scientific and VIRGO collaborations, et al. *The basic physics of the binary black hole merger GW150914.*, Annalen der Physik 529.1-2 (2017): 1600209.

8. G. R. Fowles, *Introduction to modern optics,* p. 63-65

9. Black, Eric D., and Ryan N. Gutenkunst., *An introduction to signal extraction in interferometric gravitational wave detectors,* American Journal of Physics 71.4 (2003): 365-378.

10. Richard O. Duda, Peter E. Hart. *Use of the hough transformation to detect lines and curves in pictures*, Commun. ACM, 15(1):11–15, (1972).4

11. F. Antonucci, P. Astone, S. D'Antonio, S. Frasca, C. Palomba, *Detection of periodic gravitational wave sources by Hough transform in the f versus f' plane,* Class. Quantum Grav. 25 (2008)

12. A. Liddle, P. Mukherjee, D. Parkinson, *Model selection in cosmology* (2006)