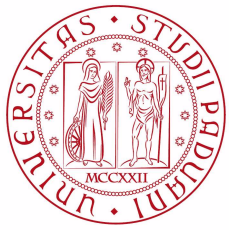

UNIVERSITÀ DEGLI STUDI DI PADOVA

Dipartimento di Fisica e Astronomia "Galileo Galilei"

Corso di Laurea Triennale in Astronomia

Tesi di Laurea

Modellazione di un interferogramma in luce parzialmente coerente

Relatore

Prof. Stefano Ciroi

Correlatori

Dr. Claudio Pernechele

Dr. Daniela Fantinel

Laureando

Paolo Cerpelloni

Anno Accademico 2018/2019

Sommario

In questa tesi si descriveranno i concetti basilari dell'interferometria a bassa coerenza e se ne esporranno alcune sue applicazioni in ambito astronomico.

Successivamente verrà esposto un codice di programmazione in linguaggio Python scritto per analizzare alcuni interferogrammi generati da un dispositivo ricostruito in laboratorio. Per mezzo di questo codice si ricaveranno alcuni valori utili per lo studio dell'interferometro quale misuratore di distanza (range finder).

In questa configurazione l'interferometro serve per controllare la distanza fra due elementi ottici, parametro essenziale per il loro allineamento.

Con questo lavoro si cerca di stabilire quale sia l'accuratezza nella misura di distanza utilizzando un interferometro a bassa coerenza in condizioni realistiche. I metodi esposti per trovare il valore dell'accuratezza della misura possono essere utili per le diverse applicazioni in cui si può utilizzare l'interferometro a bassa coerenza.

Indice

Elenco delle figure

1 L'interferenza

L'interferenza è un fenomeno fisico che avviene quando due onde si sovrappongono in un punto P dello spazio. Assumendo che le onde siano armoniche e abbiano la stessa frequenza ν, le caratteristiche della loro sovrapposizione in P dipendono dalle sorgenti e dalla direzione di osservazione del fenomeno. Le equazioni di due onde sferiche luminose emesse da due sorgenti S_1 e S_2 che si propagano verso il punto P sono:

$$E_1 = \frac{E_0}{r_1} \sin(kr_1 - \omega t + \phi_1) \tag{1}$$

$$E_2 = \frac{E_0}{r_2} \sin(kr_2 - \omega t + \phi_2) \tag{2}$$

Il termine $kr_i - \omega t$ descrive la propagazione dell'onda luminosa di ciascuna sorgente verso il punto P; il termine ϕ_i è una caratteristica intrinseca di ogni sorgente. Si definisce differenza di fase tra due onde in un punto P la quantità :

$$\delta = (kr_2 - \omega t + \phi_2) - (kr_1 - \omega t + \phi_1) = k(r_2 - r_1) + (\phi_2 - \phi_1) \tag{3}$$

Nel termine δ, differenza di fase, sono presenti due componenti: una differenza di fase intrinseca $\Delta\phi = \phi_2 - \phi_1$, che dipende dalle proprietà delle sorgenti e la componente $k(r_2 - r_1)$, che dipende dalla differenza dei percorsi $\Delta r = r_2 - r_1$. Se la differenza di fase tra due onde è costante le due sorgenti vengono definite coerenti, altrimenti vengono dette incoerenti. L'intervallo di tempo in cui si può rappresentare come una sinusoide l'onda è detto intervallo temporale di coerenza; maggiore è il tempo di coerenza, maggiore è la coerenza temporale della sorgente. La lunghezza di coerenza è invece l'intervallo spaziale in cui la sorgente oscilla in modo regolare. Due sorgenti devono essere coerenti affinché il fenomeno dell'interferenza abbia luogo.

Si prendano come esempio le sorgenti di luce ordinaria, formate da molti atomi che, oscillando con un periodo $T = 2 \cdot 10^{-15} s$, emettono onde luminose. L'emissione di un atomo avviene in un lasso di tempo pari a $\Delta t = 10^{-8} s$, in cui viene emesso un impulso di luce, un pacchetto d'onde, in cui sono presenti $N = \Delta t/T = 5 \cdot 10^6$ oscillazioni. Quindi la lunghezza spaziale di questo pacchetto è $c\Delta t = 3\ m$. Durante il tempo t la direzione di E e la fase restano costanti. Un altro atomo si diseccita in modo non correlato all'atomo precedente, emettendo un pacchetto con fase diversa dalla precedente. L'onda emessa da una sorgente di luce ordinaria è il risultato dei pacchetti d'onda dei singoli atomi e le fasi sono distribuite in maniera casuale. Il fenomeno dell'interferenza non avrà perciò luogo per sorgenti di luce ordinaria, a differenza di altre sorgenti di altra natura come LASER (Light Amplification by Stimulated Emission of Radiation) o SLED (Superluminescent Laser Diode). In particolare poi se $\Delta\phi = 0$ le due sorgenti si dicono sincrone.

Da *Elementi di Fisica: Elettromagnetismo e Onde* [Mazzoldi et al., 2012].

2 L'esperimento di Young

Il primo dispositivo con cui si riuscì a produrre l'interferenza fu costruito da Young nel 1801. Un fascio di luce ordinaria monocromatica incide su una lastra A sulla quale è stata aperta una fenditura S_0 molto sottile. La luce uscendo dalla fenditura viene diffratta e arriva su uno schermo B sul quale sono poste altre due fenditure S_1 e S_2 molto sottili, parallele a S_0. I pacchetti d'onda che escono da queste ultime due sorgenti hanno le stesse caratteristiche, quindi S_1 e S_2 sono due sorgenti coerenti sincrone. Questo metodo per ottenere sorgenti coerenti si chiama divisione del fronte d'onda. Le due onde successivamente arrivano su uno schermo C posto ad una distanza L molto grande rispetto alla separazione fra le due fenditure. Sullo schermo appare una figura detta figura di interferenza ed è formata da frange chiare e scure, queste frange sono chiamate frange di interferenza. Le frange chiare riproducono i massimi d'intensità, quelle scure i minimi. Sull'asse del dispositivo è localizzata una frangia chiara, la frangia centrale.

Siano E_1 e E_2 l'effetto prodotto dalle singole sorgenti nel punto P:

$$E_1 = \frac{E_0}{r_1}\sin(kr_1 - \omega t) \tag{4}$$

$$E_2 = \frac{E_0}{r_2}\sin(kr_2 - \omega t) \tag{5}$$

La differenza di fase

$$\delta = k(r_2 - r_1) \tag{6}$$

Nella condizione dell'esperimento di Young, il punto P si trova ad una distanza r dal punto medio delle due fenditure molto maggiore rispetto alla separazione fra le due fenditure, perciò si possono ritenere le direzioni di propagazione parallele e si può scrivere

$$r_2 - r_1 = d\sin\theta \tag{7}$$

e dalle equazioni 4, 5, 6, si ottiene

$$\delta = kd\sin\theta = \frac{2\pi}{\lambda}d\sin\theta \tag{8}$$

Il campo elettrico in P è descritto da:

$$E_p = \frac{E_0}{r}\sin(kr_1 - \omega t) + \frac{E_0}{r}\sin(kr_2 - \omega t) \tag{9}$$

L'ampiezza del campo elettrico in P è:

$$E_p = \left[\left(\frac{E_0}{r}^2\right) + \left(\frac{E_0}{r}^2\right) + 2\left(\frac{E_0}{r}^2\right)\cos\delta\right]^{1/2} \tag{10}$$

L'intensità è direttamente proporzionale al quadrato dell'ampiezza. Perciò da (10) l' intensità in P è:

$$I_p = 2I_1(1+\cos\delta) = 4I_1\cos^2\left(\frac{\delta}{2}\right) \tag{11}$$

Per cui l'intensità in funzione dell'angolo, da (8):

$$I_p(\theta) = 4I_1\cos^2\frac{\pi d\sin\theta}{\lambda} \tag{12}$$

La posizione delle frange sulla figura d'interferenza dipende dal seno dell'angolo. Si osserverà un massimo di interferenza quando la differenza $d\sin\theta$ è un multiplo intero della lunghezza d'onda, in questa condizione le onde sono in fase e l'interferenza è costruttiva.

$$\delta = \frac{2\pi}{\lambda}d\sin\theta = 2m\pi,\ d\sin\theta = m\lambda,\ m = 0,\pm1,\pm2... \tag{13}$$

Invece nei punti in cui la differenza di percorso è un multiplo semintero di λ, le onde sono in opposizione di fase e l'interferenza è di tipo distruttivo.

$$\delta = \frac{2\pi}{\lambda}d\sin\theta = (2m+1)\pi,\ d\sin\theta = (2m+1)\lambda,\ \frac{\lambda}{2}m = 0,\pm1,\pm2... \tag{14}$$

Nell'ipotesi in cui θ sia piccolo si può scrivere $\sin\theta \approx \tan\theta \approx x/L$, le equazioni 12, 13, 14 cambiano:

$$I(x) = 4I_1\cos^2\frac{\pi d}{\lambda}\frac{x}{L} \tag{15}$$

$$\theta = m\frac{\lambda}{d},\ x = m\frac{\lambda}{d},\ m = 0,\pm1,\pm2... \tag{16}$$

$$\theta = (2m+1)\frac{\lambda}{2d},\ x = (2m+1)\frac{\lambda}{2d},\ m = 0,\pm1,\pm2... \tag{17}$$

Queste equazioni descrivono quindi la figura d'interferenza, caratterizzata da massimi e minimi, che si forma sullo schermo. Le frange di interferenza possono quindi essere visualizzate come in Fig. 1.

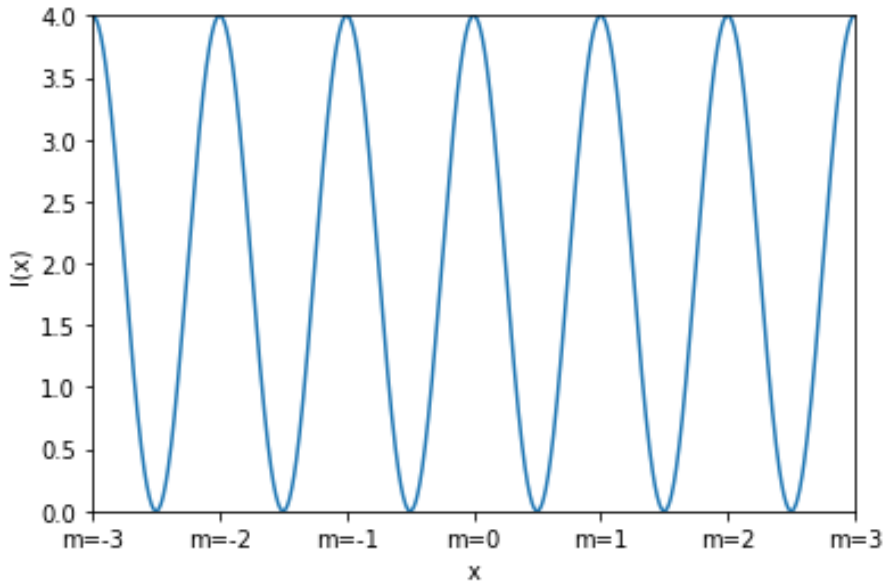

Figura 1: Figura di interferenza.

In realtà la figura di interferenza viene modulata dall'effetto della diffrazione e quindi l'intensità delle frange, da quella della frangia centrale che è massima, decresce. L'intera figura prodotta sullo schermo dall'esperimento di Young è descritta quindi da

$$\beta = \frac{\pi}{\lambda} b \sin\theta \tag{18}$$

$$\gamma = \frac{\pi}{\lambda} d \sin\theta \tag{19}$$

$$I = 4I_1 \frac{\sin^2\beta}{\beta^2} \cos^2\gamma \tag{20}$$

Nelle equazioni 18, 20 b rappresenta la larghezza di ciascuna fenditura.

Quello che si osserva quindi sullo schermo è, come detto in precedenza, il prodotto della figura di interferenza e di quella di diffrazione, come mostrato in Fig. (2).

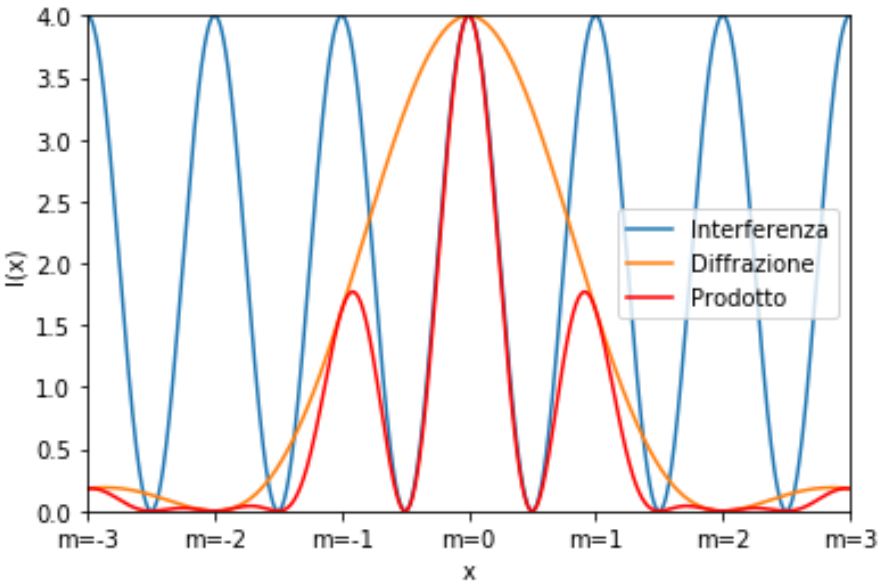

Figura 2: Figura osservata sullo schermo con l'esperimento di Young.

Da *Elementi di Fisica: Elettromagnetismo e Onde* [Mazzoldi et al., 2012].

3 Interferometri a divisione di ampiezza

Quando si parla di interferometri a divisione di ampiezza, si indicano quegli strumenti che ottengono l'interferenza per mezzo di un processo di suddivisione di un'onda incidente in due onde riflesse, le sorgenti coerenti separate in profondità. L'interfererometro a divisione di ampiezza più famoso è sicuramente quello di Michelson, ma prima di descriverne la configurazione si espongono di seguito i principi alla base del suo funzionamento.

3.1 Frange di uguale inclinazione

Supponiamo che un fascio di luce incida con un certo angolo θ_i su di un vetro trasparente di indice di rifrazione n e spessore d. La luce nel punto di contatto col vetro crea un'onda riflessa ed un'onda trasmessa con un angolo θ_t all'interno del vetro. Nel punto inferiore del vetro l'onda trasmessa crea un'onda riflessa che si propaga nuovamente all'interno del vetro. Nel punto superiore del vetro si forma un'onda trasmessa in aria con la stessa direzione della prima onda riflessa. Le altre onde riflesse e trasmesse hanno un'intensità notevolmente inferiore rispetto a queste prime due. Si creano così, per mezzo della riflessione sul vetro, due sorgenti virtuali: queste sono coerenti perché sono state prodotte dalla stessa sorgente iniziale. La differenza di fase fra le due sorgenti è:

$$\delta = 2dkn\cos\theta_t - \pi \tag{21}$$

dove $k = 2\pi/\lambda$; il primo termine a secondo membro descrive la differenza di fase dovuta alla differenza di percorso effettuato dalle due onde; il termine π la differenza di fase prodotta alla riflessione.

Se poniamo uno schermo ad una distanza molto grande rispetto allo spessore del vetro o guardando verso lo spessore superiore del vetro o raccogliendo la luce per mezzo di una lente, si osserva una figura di interferenza. I raggi paralleli riflessi dalla faccia superiore del vetro vanno a creare la figura di interferenza in una posizione che dipende dalla loro inclinazione. La figura ha luogo anche se la sorgente è estesa. Le frange di questa figura di interferenza si chiamano frange di uguale inclinazione. La fase può essere riscritta così:

$$\delta = \frac{4\pi nd\cos\theta_t}{\lambda} - \pi \tag{22}$$

I massimi sono descritti in questo modo:

$$\delta = 2m\pi,\ d\cos\theta_t = (2m+1)\frac{\lambda}{4n},\ m = 0, 1, 2, ... \tag{23}$$

I minimi in quest'altro:

$$\delta = 2(m+1)\pi,\ d\cos\theta_t = 2m\frac{\lambda}{4n},\ m = 0, 1, 2, ... \tag{24}$$

Se l'illuminazione e l'osservazione avvengono con l'asse parallelo al vetro, le frange assumono la forma di anelli. Queste frange sono chiamate frange di Haidinger.

Figura 3: Frange di Haidinger. Da *Dispense del Corso di Ottica Applicata* [D'Onofrio, 2014].

Da *Elementi di Fisica: Elettromagnetismo e Onde* [Mazzoldi et al., 2012] e *Elementi di Ottica per Astronomi* [D'Onofrio, 2004].

3.2 Frange di uguale spessore

Le frange di uguale spessore derivano da una variazione di spessore del vetro. Se il vetro possiede la forma di un cuneo, le due facce piane formano tra loro un angolo α. Ad ogni punto del cuneo è associato uno spessore $d = \alpha x$, con α piccolo. Facendo incidere, quasi perpendicolarmente, una sorgente estesa sul cuneo, si osserva la figura di interferenza. Come altro esempio al posto di un cuneo di vetro si possono utilizzare due lamine separate da un angolo α. I massimi delle frange seguono le seguenti equazioni:

$$d = (2m+1)\frac{\lambda}{4n},\ x = (2m+1)\frac{\lambda}{4n\alpha},\ m = 0, 1, 2, ... \tag{25}$$

Per i minimi si ha:

$$d = m\frac{\lambda}{2n},\ x = m\frac{\lambda}{2n\alpha},\ m = 0, 1, 2, ... \tag{26}$$

Sulla superficie del cuneo si osservano frange chiare e scure, ovvero le frange di uguale spessore, caratterizzate ognuna da un valore dello spessore d. Per $x = 0$,

quindi $d = 0$, ovvero sul bordo del cuneo, la prima frangia è scura, è un minimo; infatti dalle (22) e (11) l'intensità è zero.

Da *Elementi di Fisica: Elettromagnetismo e Onde* [Mazzoldi et al., 2012].

3.3 L'interferometro di Michelson

L'interferometro di Michelson, come menzionato in precedenza, è un interferometro a divisione di ampiezza. La sua configurazione è illustrata in figura 4. Una sorgente di luce laser puntiforme viene collimata per mezzo di un beam expander, ovvero un sistema formato da lenti con lo scopo di aumentare il raggio della sorgente puntiforme e di renderne paralleli i raggi luminosi. Successivamente il fascio incide su un beam splitter, un dispositivo che ha il compito di dividere in ampiezza il fascio incidente. Il beam splitter è una lamina di vetro posizionata a 45 gradi rispetto al fascio, di cui una faccia è semiriflettente. Con questo meccanismo il fascio è diviso in due. Un fascio è diretto verso un specchio M1 posto in asse alla sorgente, l'altro verso uno specchio mobile M2 posto perpendicolarmente alla sorgente. I due fasci vengono riflessi dagli specchi e tornano verso la parte semiriflettente del beam splitter, dove vengono rispettivamente trasmessi e riflessi. Per mezzo di una lente vengono fatti convergere su di un detector dove si crea la figura di interferenza. La differenza di cammino tra le onde dipende esclusivamente $d = d_2 - d_1$, dove d_1 e d_2 sono le distanze degli specchi dal beam splitter. Se i due specchi sono perpendicolari tra loro, l'effetto è lo stesso di quello di una lamina d'aria di spessore $d = d_2 - d_1$ e perciò si osserveranno le frange circolari o di Haidinger. La frange sono osservabili poiché i due raggi percorrono un differente cammino ottico. Il raggio che va verso lo specchio M2 passa per il beam splitter tre volte mentre quello verso M1 una volta: questo crea una differenza di fase di π. Se al posto di luce monocromatica, come quella laser, si usasse luce ordinaria bisognerebbe inserire nel tratto di percorso dal beam splitter allo specchio M1 una lastra compensatrice. Questa lastra permette ai raggi di luce ordinaria che interferiscono di attraversare lo stesso spessore di vetro e quindi di evitare dispersioni dovute all'indice di rifrazione, poiché quest'ultimo dipende da λ. Un aspetto molto importante da sottolineare è che la figura di interferenza è osservabile soltanto se la differenza di cammino ottico delle onde tra i due specchi è minore della lunghezza di coerenza della sorgente.

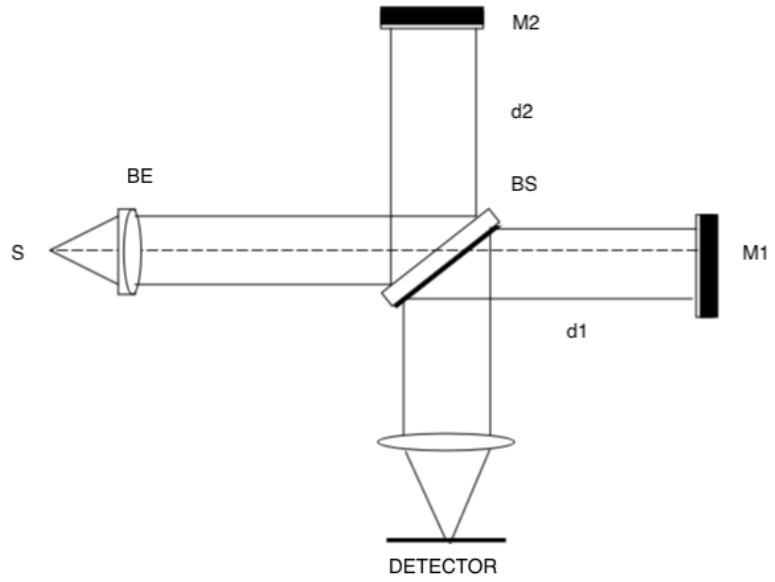

Figura 4: Interferometro di Michelson. $S = sorgente$; $BE = beam\ expander$, $BS = beam\ splitter$, $M1 = specchio\ 1$, $M2 = specchio\ 2$, $d1 = distanza\ dal\ beam\ splitter\ allo\ specchio\ 1$, $d2 = distanza\ dal\ beam\ splitter\ allo\ specchio\ 2$. Da "*Elementi di ottica per astronomi*" [D'Onofrio, 2004].

Per descrivere quindi le frange si possono utilizzare le equazioni 23, 24. Le frange appaiono come anelli chiari e scuri, ad ogni anello è associato un ordine m. Muovendo M2 verso il beam splitter, d decresce e secondo la (23) $cos(\theta_t)$ cresce e θ_t decresce. Dopo ogni spostamento di M2 se d decresce di $\lambda/2$, gli anelli si concentrano verso il centro e gli ordini m alti scompaiono. Gli anelli rimanenti si allargano. La frangia centrale, se $d = 0$, occupa tutto lo schermo e a causa della differenza di fase di π sarà scura. Continuando a muovere lo specchio M2 le frange ricompaiono. (Fig. 3).

Se i due specchi M1 e M2 sono inclinati tra loro si possono osservare le frange di uguale spessore; i due specchi possono essere rappresentati come due lamine che formano un cuneo separate da uno spessore d'aria. Si possono quindi utilizzare le eq 25, 26 per descrivere le frange. La distanza tra due frange chiare successive è equivalente alla differenza di un λ. Per calcolare l'angolo α tra i due specchi si misura la distanza tra due massimi successivi e poi si utilizza la seguente relazione:

$$\sin \alpha = \frac{\lambda}{2x} \tag{27}$$

Da *Elementi di Fisica: Elettromagnetismo e Onde* [Mazzoldi et al., 2012] e *Elementi di Ottica per Astronomi* [D'Onofrio, 2004].

4 Interferometro a bassa coerenza

L'interferometro a bassa coerenza o LCI (Low Coherence Interferometer) è un dispositivo che utilizza una sorgente a bassa coerenza (luce bianca). Si costruisce un interferometro di Michelson in cui la sorgente è un superluminescent laser diode (SLED). La caratteristica di questa sorgente luminosa è quella di avere una lunghezza di coerenza bassa. A causa di questa proprietà la figura di interferenza si osserverà soltanto se la distanza d, ovvero la differenza tra la distanza dei due specchi, è all'interno della lunghezza di coerenza della sorgente, tipicamente alcune decine di μm per uno SLED. Perciò l'interferogramma avrà la tipica forma di cosinusoide convoluta con una gaussiana e viene risolto il problema dell'ambiguità di fase tipica degli interferometri che utilizzano un laser come sorgente. I fronti d'onda vengono ricombinati per mezzo di una lente su un sensore bi-dimensionale (nel dispositivo replicato per ottenere i dati da analizzare per questa tesi è stato usato un sensore CCD). L' LCI viene descritto da alcune equazioni che prendono in considerazione anche la sorgente a bassa coerenza. La lunghezza di coerenza della sorgente è descritta dalla seguente relazione:

$$L_c = \frac{2\ln 2}{\pi}\frac{\lambda_0^2}{\Delta\lambda_{\frac{1}{2}}} \tag{28}$$

dove λ_0 è la lunghezza d'onda caratteristica della SLED, $\Delta\lambda_{\frac{1}{2}}$ è la larghezza di banda.

L'intensità della figura di interferenza ideale è caratterizzata dalla componente data dall'interferenza ($cos^2()$) e da una componente gaussiana con una larghezza caratterizzata dalla lunghezza di coerenza della SLED. La derivata della figura di interferenza è:

$$I(\Delta D) \propto A\exp\left(-\frac{\pi^2}{\ln 2}\frac{\Delta D^2}{L_c^2}\right)\cos\left(\frac{4\pi}{\lambda_0}\Delta D\right) \tag{29}$$

$I(\Delta D)$ è l'intensità della figura sul rivelatore, ΔD corrisponde alla differenza di cammino ottico fra i due bracci d_2 e d_1 dell'interferometro, A è l'ampiezza. Il modello delle frange che si possono osservare è formato quindi dalla funzione coseno $\cos\left(\frac{4\pi}{\lambda_0}\Delta D\right)$ e da una funzione gaussiana $\exp\left(-\frac{\pi^2}{\ln 2}\frac{\Delta D^2}{L_c^2}\right)$ che modula la funzione coseno. La forma ideale della derivata della figura di interferenza prodotta da un interferometro a bassa coerenza è illustrata in Fig. 5.

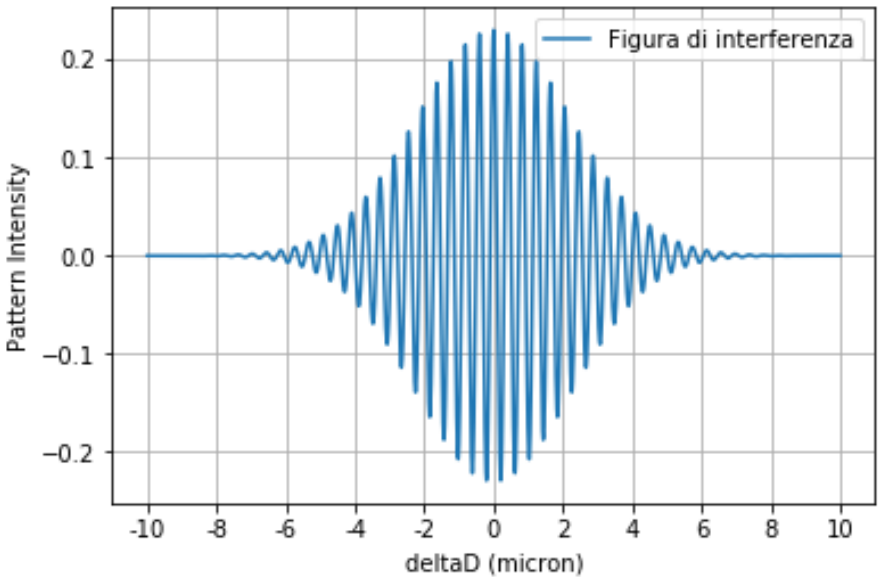

Figura 5: Profilo della derivata della figura di interferenza prodotta da un interferometro in configurazione di Michelson e da una sorgente SLED.

Il massimo delle frange corrisponde alla posizione in cui i cammini ottici dei due bracci sono uguali.

Da *Low coherence interferometry-based meter-distance range finder* [Pernechele et al., 2017a], *Positioning of Optical Payload – SALT Telescope* [Courteville et al., 2004], *Optical plate thickness measurement with single-shot low coherence interferometry* [Pernechele et al., 2017b] e *Non-contact in-process metrology using a high-accuracy low-coherence interferometer* [Courteville et al., 2005].

4.1 Applicazioni dell'interferometro a bassa coerenza

L'interferometro a bassa coerenza può essere utilizzato in vari ambiti, di seguito vengono esposte alcune applicazioni.

4.1.1 Il range finder

Il range finder è uno strumento in grado di misurare la distanza di un oggetto con una precisione sub-micrometrica e la variazione in profondità di qualche millimetro. Per ottenere questi risultati lo strumento presenta alcune differenze dalla classica configurazione di Michelson. Lo specchio mobile, presente su un

braccio dell'interferometro, viene sostituito con l'oggetto da analizzare. Al posto dell'altro specchio, quello fisso, viene posto uno *stepped mirror*, uno specchio composto da scalini; il fronte d'onda riflesso da questo dispositivo risulta tiltato rispetto al fronte d'onda dell'oggetto. Lungo il braccio dove è presente l'oggetto, viene posta una lente, in tal modo solo un singolo punto dell'oggetto viene ripreso. La combinazione dei due fronti d'onda produce, sul sensore (un CMOS), l'interferogramma, la cui posizione lungo il rivelatore è determinata dalla distanza dell'oggetto. Il sistema non presenta parti mobili poiché la sorgente, il sensore, il CMOS e lo stepped mirror sono fissi; la distanza dell'oggetto è la sola parte che varia; questa caratteristica è utile per applicazioni metrologiche spaziali dove la presenza di parti in movimento rappresenta una possibile causa di fallimento del sistema. Lo stepped mirror è descritto da diversi parametri: il numero di scalini presenti, la loro lunghezza, larghezza e profondità. L'intervallo di profondità con cui è possibile misurare è determinato dalla profondità del singolo scalino per il numero di scalini. L'interferogramma si forma in corrispondenza di un determinato scalino e cambia posizione lungo l'immagine dello stepped mirror se la distanza dell'oggetto varia. Si può quindi con questo strumento calcolare anche lo spessore ottico di un pezzo di vetro. Infatti il fronte d'onda viene riflesso dalla prima superficie del vetro e poi, dopo aver attraversato lo spessore del vetro, dalla seconda superficie del vetro. Si creano due figure di interferenza relative alle due superfici. La loro posizione sul sensore è diversa. Conoscendo la larghezza degli scalini si misura la distanza tra i due interferogrammi, che corrisponde al cammino ottico. Sapendo poi l'indice di rifrazione n del vetro utilizzato, si ricava lo spessore del pezzo di vetro. Da *Low coherence interferometry-based meter-distance range finder* [Pernechele et al., 2017a].

Il range finder può anche essere utilizzato in ambito astronomico. Questi stessi concetti vengono utilizzati per uno strumento, la cui configurazione presenta alcune variazioni rispetto a quella esposta in precedenza, costruito nell'ambito della missione PLATO, il cui principale obiettivo scientifico è la scoperta e lo studio di sistemi planetari extrasolari attraverso la rivelazione di transiti planetari, da *Low coherence interferometry-based meter-distance range finder* [Pernechele et al., 2017a] . Un range finder viene anche utilizzato per la calibrazione dello strumento MICADO per E-ELT(European Extremely Large Telescope). MICADO è progettato per avere alte prestazioni nell'ottenere immagini con accuratezza del micro-arcsec. Per calibrare lo strumento è stata sviluppata una maschera di calibrazione per rimuovere dallo strumento le distorsioni ottiche. Sulla maschera sono presenti diversi pinhole che, se illuminati, si comportano come sorgenti puntiformi. La posizione dei fori sulla maschera va conosciuta con esattezza per avere una calibrazione affidabile. Il valore della separazione tra i pinhole è ottenuta misurando la distanza fra le frange su uno schermo, utilizzando gli stessi principi dell'esperimento di Young. Tuttavia, per avere delle coordinate affidabili per i fori, la distanza tra lo schermo e la maschera deve essere conosciuta con la precisione del micron. Si utilizza perciò un range finder. Lo stepped mirror viene posizionato sulla stessa struttura della maschera, da *Low coherence interferometry-based meter-distance range finder* [Pernechele et al., 2017a] e

The MICADO first light imager for the ELT: preliminary design of the MICADO Calibration Assembly [Rodeghiero et al., 2018].

4.1.2 Profilometro LCI

Anche questo interferometro a bassa coerenza è costruito sullo schema di Michelson. L'unica differenza con lo schema classico è che lo specchio mobile è sostituito dall'oggetto da analizzare. Il fascio uscente dal beam splitter passa attraverso un obiettivo e arriva sul rivelatore un CMOS. L'oggetto è fisso, mentre il sistema (composto da sorgente, collimatore, beam splitter, specchio, obiettivo e detector) si può muovere lungo la direzione di scansione, cioè quella perpendicolare all'oggetto. Quando l'interferogramma appare sul sensore si determina la posizione con una precisione di frazioni di micron. Durante la scansione un'immagine viene presa ogni 0.1 μm. In tal modo per l'oggetto si determina la posizione di ogni elemento spaziale e si ricostruisce l'immagine tridimensionale della superficie. Un vantaggio di questo strumento è quello di non entrare in contatto con l'oggetto da analizzare. Da *3D surface measurements with Low Coherence Interferometry* [Pernechele and Chinellato,] e *Multi-thread real- time data processing for an image-based partially coherent light interferometer* [Pernechele et al., 2009].

Le applicazioni di questo strumento sono molteplici, da quelle astronomiche fino a quelle per la ricostruzione della superficie della tela dei dipinti. Un esempio è la determinazione della curvatura e della profondità ottica di un doppietto acromatico. Uno strumento di questo tipo è stato utilizzato per determinare con precisione la posizione delle antenne dello strumento LFI (Low Frequency Instrument) presente sul piano focale del telescopio Planck. Lo scopo della missione ESA Planck è stato ricavare una mappa di tutto il cielo della radiazione cosmica di fondo (CMB). Da *Planck LFI flight model feed horns* [Villa et al., 2010].

Un interferometro in questa configurazione può essere utilizzato anche per misurare il profilo di pannelli radio. Il Sardinia Radio Telescope (SRT) è composto da un' antenna di 64 metri di diametro e lavora ad una lunghezza d'onda di 3 mm. La superficie totale dell'antenna primaria è composta da pannelli di 1/2 metri che vengono installati sulla struttura principale. Questi pannelli devono avere specifiche molto stringenti per non influenzare negativamente la qualità del radio telescopio. Per questo motivo i pannelli vengono esaminati con il profilometro ottenendo valori con una precisione di ± 10 μm. Da *Surface measurements of radio antenna panels with white-light interferometry* [Chinellato et al., 2010].

5 Modellazione di un interferogramma in luce parzialmente coerente

In questa sezione si descrive la modellazione di un interferogramma in luce parziamente coerente mediante l'acquisizione dei dati. Per la raccolta dei dati si è costruito in laboratorio un interferometro a bassa coerenza in configurazione di Michelson. Per la sorgente si è utilizzato uno SLED. È stato utilizzato il modello EXS8310-B001 della della marca EXALOS. La lunghezza d'onda è compresa fra 800 nm e 840 nm e la lunghezza d'onda centrale è $\lambda_0 = 820\ nm$, la larghezza di banda $\Delta\lambda_{\frac{1}{2}} = 25\ nm$ e la lunghezza di coerenza $L_c = 12\ \mu m$. Il fronte d'onda passa attraverso un collimatore con lunghezza focale $f = 36.18\ mm$, il modello F810APC-842 prodotto dalla THORLABS. Il fascio arriva al beam splitter dove viene diviso per poi essere riflesso sullo specchio M1 e sullo specchio M2. Il beam splitter è un Cage Cube-Mounted Non-Polarizing Beamsplitter modello CCM1-BS014/M prodotto dalla THORLABS. I due fronti d'onda dopo essersi ricombinati sul beam splitter incontrano una lente. Questa lente è un obiettivo telecentrico con un ingrandimento $I = 0.385$. Il modello utilizzato è il TC12016 della OPTO ENGINEERING. Il fascio poi arriva sul rivelatore. Il rivelatore utilizzato è un CCD modello PL-B957 prodotto dalla PIXELINK. La dimensione del sensore è 1392 $\times$ 1040 pixel per lato. La dimensione del pixel è 6.45 μm.

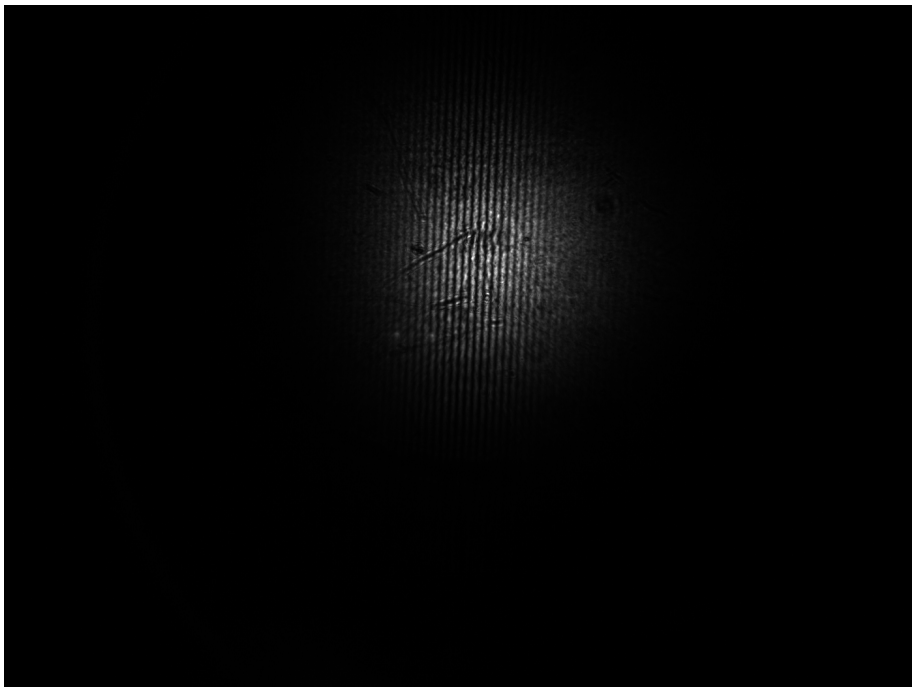

Figura 6: Immagine delle frange di interferenza per l'interferometro realizzato per la raccolta dei dati.

Dopo aver spostato adeguatamente uno specchio per fare in modo che la distanza tra i due sia minore della lunghezza di coerenza della SLED si osserva la figura di interferenza. I due specchi sono tra loro tiltati, le frange che si potranno osservare saranno quelle di eguale spessore. Si procede quindi ad ottenere venti immagini della figura di interferenza; queste sono la base su cui si lavora. Le immagini vengono salvate dal sensore in formato binario non compresso, *.bin*. Questa estensione ha il vantaggio di avere dimensioni in media più ridotte e permette una più facile modifica del file, possiede però lo svantaggio di non mantenere la portabilità da un calcolatore ad un altro. In Fig. 6 una delle immagini prese in laboratorio.

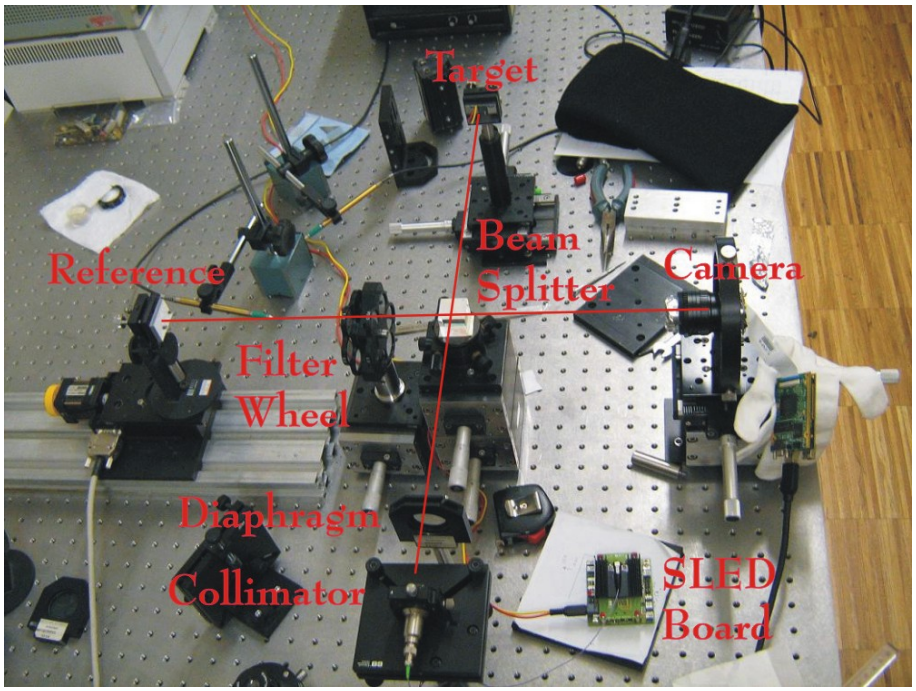

Figura 7: Stessa configurazione dello strumento costruito in laboratorio.

Per l'analisi dei dati si è utilizzato Anaconda, una distribuzione open source per la gestione dei pacchetti software, delle dipendenze e degli ambienti del linguaggio Python. Anaconda Navigator, l'interfaccia grafica, permette l'installazione dei pacchetti utili all'analisi dei dati e anche di lanciare alcune applicazioni. Una di queste è Spyder che fornisce un ambiente di sviluppo per la programmazione scientifica che utilizza il linguaggio Python. La versione Python utilizzata è la 2.7. La procedura comprende due files: un file chiamato *libreria.py* per le funzioni e il file *main.py*. Le funzioni che verranno chiamate per ricavare i risultati sono $find\ angle$ e *main* definite nel file *main.py*. Per poter utilizzare queste funzioni e quelle di cui sono composte bisogna caricare i file *libreria.py* e *main.py* con il comando $runfile$ (in questo caso se si sta usando Spyder). In tal modo si avranno a disposizione le funzioni sulla shell.

L'obiettivo principale di queste funzioni è quello di ricavare il valore dell'incertezza sulla posizione relativa degli specchi nei due bracci dell'interferometro, ovvero della differenza fra le loro lunghezze, tramite due metodi differenti. Questi valori rappresentano l'errore con cui si conosce l'esatta posizione di uno specchio. Come esempio consideriamo una scansione di un oggetto per mezzo di un profilometro (sezione 4.1.2). La scansione può acquisire un'immagine dell'oggetto ogni decimo di micron; i valori calcolati con le funzioni esposte in questa

tesi si riferiscono all'errore associato alla posizione di un'immagine durante la scansione.

Il primo metodo consiste nell'ottenere il valore dell'incertezza tramite il valore di spostamento medio ottenuto per mezzo della correlazione tra un grafico derivata della figura di interferenza scelto come riferimento e gli altri grafici della derivata della figura di interferenza. Si è utilizzata la derivata della figura di interferenza in modo da isolare il segnale della luminosità di fondo presente sulla figura di interferenza.

Il secondo metodo ricava il valore dell'incertezza utilizzando il valore dello spostamento calcolato con la differenza tra un valore della fase di un fit della derivata della figura di interferenza e i valori delle fasi degli altri fit.

5.1 La procedura main

Tramite la procedura *main* si esegue l'analisi dei dati da cui si ricavano diversi valori utili per lo studio dell'interferometro a bassa coerenza. In questo caso sono state prese in laboratorio 20 immagini delle figure di interferenza con lo strumento fermo nella stessa posizione. Per prima cosa la funzione ruota le immagini poiché le frange di interferenza sono inclinate di un certo angolo come si vede dalla figura 6. Questo è dovuto a un disallineamento residuo tra gli specchi.

Dopo aver ruotato le immagini si ricava per ognuna di esse la figura di interferenza e successivamente la derivata della figura di interferenza. Si utilizza la derivata della figura di interferenza per pulire il possibile rumore di fondo presente sulla figura di interferenza e ottenere quindi una curva più semplice da analizzare.

Si ricavano i fit sui dati delle derivate delle figure di interferenza; la curva dei fit è descritta dell'equazione 29. Per mezzo della funzione che ricava il fit si trovano anche i parametri ottimali e i loro errori che descrivono la curva che meglio interpola i dati. Si ottiene poi la media dei diversi parametri.

I valori delle figure di interferenza sono stati catturati dal rivelatore senza muovere le componenti dello strumento. Si nota che la posizione degli interferogrammi non è la stessa per tutte le immagini. Il passo successivo è ricavare a quanto equivale lo spostamento medio degli interferogrammi rispetto ad un interferogramma di riferimento. Questi spostamenti si generano a causa di diversi errori dovuti alla natura dello strumento.

Per ottenere gli spostamenti si usano due procedimenti differenti. Il primo metodo utilizza la correlazione per ricavare gli spostamenti tra i dati della derivata di una figura di interferenza di riferimento e i dati delle altre derivate della figura di interferenza. Viene applicato lo stesso procedimento anche per calcolare il valore dello spostamento tra i fit delle derivate della figura di interferenza e il fit

di una derivata della figura di interferenza di riferimento. Per il calcolo successivo dell'incertezza viene utilizzato il valore dello spostamento relativo ai dati delle derivate delle figure di interferenza. Per ottenere un valore più accurato dello spostamento si è eseguita la correlazione con una risoluzione al di sotto del pixel, in particolare 0.01 pixel.

Il secondo procedimento consiste nell'ottenere il valore di spostamento dalla differenza tra il valore della fase di un fit di riferimento e i valori della fase degli altri fit. I valori delle fasi sono stati estrapolati dalla funzione che calcola il fit. Dal parametro $k = 4\pi/\lambda$ del fit (da (29)) si ricava la distanza tra due massimi successivi e quindi dall'equazione 27 l'angolo con cui sono inclinati tra loro gli specchi.

Una volta determinati l'angolo di inclinazione e i valori dello spostamento medio si può ricavare l'incertezza sulla posizione zero dello specchio. Si otterranno due valori uno relativo allo spostamento medio calcolato dalla correlazione e uno dallo spostamento medio calcolato dalla fase dei fit.

La funzione *main* restituisce l'angolo con cui sono tiltati gli specchi, l'errore dell'angolo con cui sono tiltati gli specchi, le due incertezze calcolate con i due diversi valori dello spostamento e i relativi errori delle incertezze. Durante l'esecuzione della funzione vengono salvati i valori trovati su dei file di testo per poterli analizzare una volta che la funzione è stata eseguita.

Di seguito si mostra lo schema della funzione *main*.

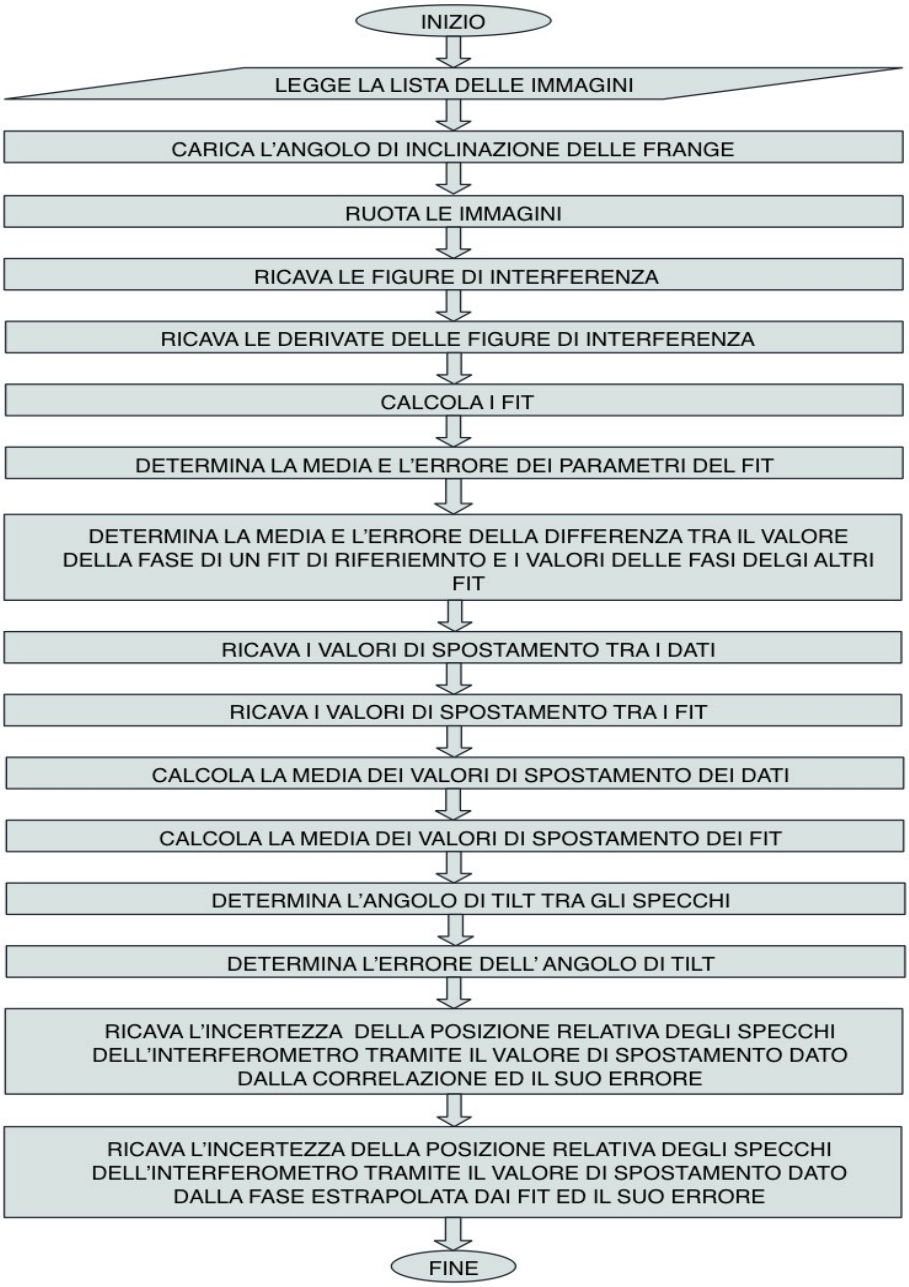

Figura 8: Schema della funzione *main*. La funzione restituisce l'angolo con cui sono tiltati gli specchi, l'errore dell'angolo di tilt, l'incertezza sulla posizione relativa degli specchi calcolata con il valore medio di spostamento ottenuto dalla correlazione, l'errore dell'incertezza e l'incertezza sulla posizione relativa degli specchi calcolata con il valore medio di spostamento ottenuto con i valori della fase estrapolati dai fit e l'errore ad essa associato.

Si presenta di seguito il codice sorgente della funzione *main*.

Listing 1: main.py, funzione *main*.

```
1
2  # Si importano le funzioni definite nel file libreria.py.
3  from libreria import*
4
5
6
7  # Input: nome lista contenente i nomi dei file binari
8
9  # Output: angolo di tilt, errore dell'angolo di tilt, incertezza sullo
       posizione relativa degli specchi calcolata con il valore di
       spostamento medio tra i dati ottenuto con la correlazione, errore
       dell'incertezza, incertezza sullo posizione relativa degli specchi
       calcolata con il valore di spostamento medio tra i fit ottenuto con
       il valore della fase, errore dell'incertezza con il valore della
       fase.
10
11
12 def main(r):
13     #Si carica l'angolo di inclinazione delle frange
14     angle = load_valore('interpolazione lineare.txt',0)
15
16     #Si ruotano le immagini
17     rot = rotazione(angle,r)
18
19     #Si ricava la figura di interferenza
20     V1_list = V1(rot)
21
22     #Si ricava la derivata della figura di interferenza
23     DATA = Vder(V1_list)
24
25     #Si ottengono i fit e i loro parametri
26     FIT = fit(DATA)
27
28     # Si caricano i valori dei parametri e si ottiene la media di sigma,
           k, D, phi e deltak. Si salva poi un file di testo.
29     Sigma = load_valori('valori_fit.txt',0)
30     sigma ,errsigma = media(Sigma)
31     K = load_valori('valori_fit.txt',1)
32     k, err_k = media(K)
33     d = load_valori('valori_fit.txt',4)
34     D, err_D = media(d) .
35     fase = find_phi(DATA)
36     np.savetxt('media parametri.txt', (sigma, errsigma,k,err_k, D, err_D,
            fase,err_fase), header = 'sigma,errore sigma, k,errore k, D,
           err_D, fase, errore fase')
37
```

```
38      #Si trovano i valori di spostamento tra i dati e tra i fit per mezzo
            della correlazione. Si salvano dei file di testo contenenti i
            valori di interesse.
39      data_correlation = find_correlation(DATA)
40      np.savetxt("data_correlation.txt", data_correlation, header = 'data
            correlation \n')
41      fit_correlation = find_correlation(FIT)
42      np.savetxt("fit_correlation.txt", fit_correlation, header = 'fit
            correlation \n')
43
44      #Si calcolano la media e la deviazione standard dei valori di
            spostamento.
45      media_data, strd_data = media(data_correlation)
46      np.savetxt("media_data.txt", (media_data,strd_data), header = 'media
            data e strd data \n')
47      media_fit, strd_fit = media(fit_correlation)
48      np.savetxt("media_fit.txt", (media_fit,strd_fit), header = 'media fit
             e strd fit \n')
49
50      #Si calcola l'angolo di tilt tra gli specchi e il suo errore.
51      tilt_angle = angle_tilt(k, 0.385, 6.45*10**-6, 820*10**-9)
52      err_tilt_angle = delta_angle_tilt(k,err_k, 0.385, 6.45*10**-6,
            820*10**-9, 1*10**-9 )
53
54      #Si calcolano i valori sull'incertezza della posizione relativa degli
             specchi e si salva un file di testo.
55      deltaD_correlazione = accuratezza(tilt_angle, media_fit, 6.45*10**-6,
             0.385)
56      deltaD_fase = accuratezza(tilt_angle, fase, 6.45*10**-6, 0.385)
57      errD_corr = delta_accuratezza(tilt_angle, err_tilt_angle,media_data,
            strd_data,6.45*10**-6, 0.385 )
58      errD_fase = delta_accuratezza(tilt_angle, err_tilt_angle,fase,
            err_fase,6.45*10**-6, 0.385 )
59      val = [tilt_angle,err_tilt_angle, deltaD_correlazione, errD_corr,
            deltaD_fase, errD_fase]
60      np.savetxt('angolo e valori accuratezza.txt',val, header='tilt_angle,
            err_tilt_angle, deltaD_correlazione, errore deltaD correlazione,
            deltaD_fase, errore DeltaD fase')
61
62      return tilt_angle,err_tilt_angle, deltaD_correlazione,errD_corr,
            deltaD_fase, errD_fase
```

Si descrivono ora singolarmente le funzioni impiegate nella procedura *main*. Queste sono contenute all'interno del file libreria.py.

Per prima cosa nel file libreria.py si importano diversi pacchetti e funzioni che verranno poi utilizzati. Questi sono:

- *numpy*: viene utilizzato per per la gestione degli array. Questa libreria permette di lavorare, come detto, con array multidimensionali ma anche con una raccolta di funzioni presenti al suo interno; queste sono di tipo matematico, logico, per la manipolazione degli array e alcune funzioni di algebra lineare.
- *matplotlib.pyplot*: è una libreria per la creazione di grafici 2D; il modulo pyplot è un'interfaccia pensata per un utilizzo interattivo di matplotlib.
- *pandas*: è una libreria che contiene strumenti per l'analisi e la modellazione dei dati.
- *OpenCv*: è una libreria con numerosi algoritmi per la lavorazione delle immagini. In particolare si è utilizzato questo pacchetto per selezionare su un 'immagine dei punti e ruotarla ma può essere usato anche per altri scopi come la traslazione di un'immagine o il tracciamento dei movimenti di un oggetto.
- *scipy*: è una libreria per il calcolo scientifico che contiene molte più funzioni di algebra lineare rispetto a numpy. In particolare da questo pacchetto si importano due funzioni:
  - *curve fit*: serve per calcolare il fit di una curva su di un set di dati.
  - *interp*1*d*: permette di interpolare funzioni uno-dimensionali.

La prima funzione che viene eseguita dalla procedura *main* è *load valore*. Questa funzione viene utilizzata per leggere un valore da un file di testo. Essa, all'inizio della procedura *main*, carica il valore dell'angolo di inclinazione delle frange; questo viene ricavato per mezzo della funzione *find angle* descritta nella sezione 5.2. *Load valore* verrà usata ancora all'interno di altre funzioni per leggere altri valori di interesse.

Listing 2: main.py, funzione *load valore*.

```
1  # Input: nome lista contenente il valore che si vuole leggere, riga della
       lista del valore che si vuole leggere.
2
3  # Output: valore
4
5  def load_valore(filetxt, e):
6
7      #Si leggono i dati dal file di testo.
8      dat = np.genfromtxt(filetxt)
9
10     #Si seleziona la riga relativa al valore.
11     valore = dat[e]
12
```

```
13      return valore
```

Una volta caricato l'angolo si ruotano le immagini in modo tale da rendere verticali le frange. Per applicare la rotazione si è scritta la funzione *rotazione*.

Listing 3: libreria.py, funzione *rotazione*.

```
1  # Input: angolo di inclinazione delle frange, nome lista contenente i
       nomi dei file binari.
2
3  # Output: lista dei valori degli array ruotati.
4
5  def rotazione(angle,r):
6      rot = []
7
8      # Si apre in modo binario r = 'lista.txt'
9      with open(r, 'rb') as c:
10
11         # Si crea una lista leggendo gli oggetti contenuti in r.
12         lista_immagini = [line.strip() for line in c]
13
14         l = len(lista_immagini)
15
16         # Si esegue un ciclo for su tutte le immagini. Ognuna di esse
               viene aperta e ne viene scambiato l'ordine dei bytes.
17         for i in range(l):
18             with open(lista_immagini[i], 'rb') as f:
19                 data = np.fromfile(f,dtype=np.uint16)
20                 swap = data.byteswap()
21                 array = np.reshape(swap, [1040,1392])
22
23                 # Si dichiarano l'altezza, la larghezza, il centro e la
                       scala dell'immagine.
24                 (h,w) = array.shape[1::-1]
25                 center = (w / 2, h / 2)
26                 scale = 1.
27
28                 # Con OpenCv si calcola la matrice di rotazione. La
                       rotazione avviene in senso antiorario e viene
                       applicata al centro dell'immagine.
29                 M =(cv2.getRotationMatrix2D(center, angle, scale))
30
31                 # Si applica la matrice di rotazione ottenendo l'immagine
                       ruotata.
32                 rotated = cv2.warpAffine(array, M, (h,w))
33
34                 # Si aggiunge l'immagine ottenuta alla lista vuota rot.
```

```
35                 rot.append(rotated)
36         return rot
```

Si mostra di seguito una delle immagini ruotate che sono state ottenute.

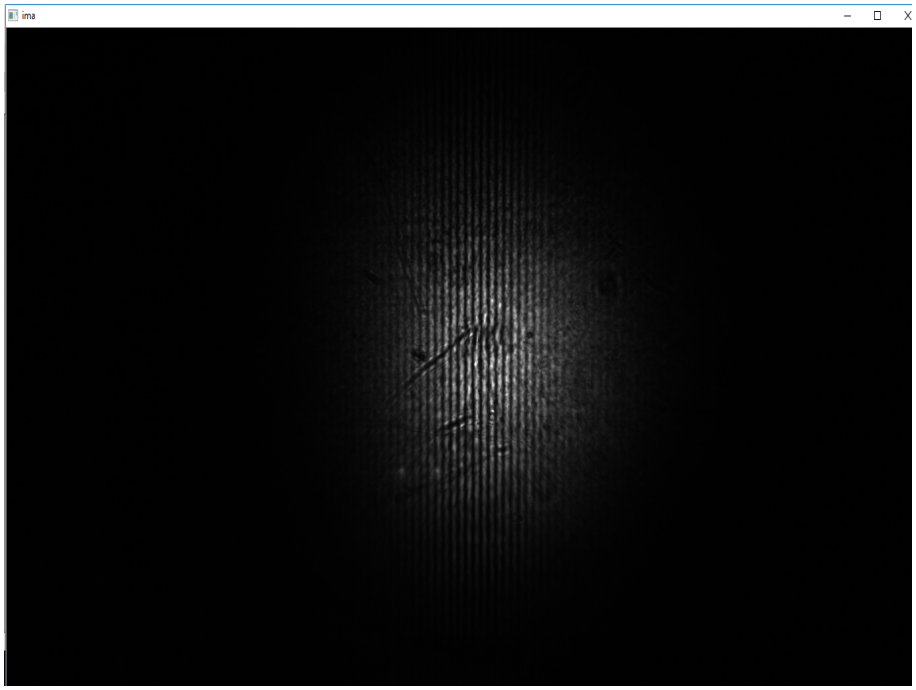

Figura 9: Immagine raddrizzata del file i-esimo.

Una volta eseguita la rotazione delle immagini, per mezzo della funzione $V1$ si calcola un vettore 1D ottenuto dalla somma di tutte le righe dell'array. Si ottiene poi il grafico della figura di interferenza.

Listing 4: libreria.py, funzione $V1$.

```
1  # Input: lista dei valori degli array ruotati.
2
3  # Output: lista dei valori dei vettori 1D.
4
5  def V1(rot):
6      V1_list = []
7      l = len(rot)
8      for i in range(l):
```

```
9
10          # Per ogni array si esegue la somma di tutte le righe e poi si
                normalizza il vettore.
11          V1_df = np.sum(rot[i], axis = 0, dtype = np.float32)
12          V1 = (V1_df - V1_df.min())/(V1_df.max() - V1_df.min())
13
14          # Si crea e si salva il grafico del vettore V1, questo è il
                grafico della figura di interferenza.
15          plt.plot(V1, label = 'V1')
16          plt.xlabel('pixel')
17          plt.ylabel('Pattern Intensity')
18          plt.title('Vettore 1D somma righe')
19          plt.legend()
20          plt.savefig('vettore_1D_somma_righe' + str(i) + '.jpg')
21          plt.close()
22
23          # Si aggiunge V1 alla lista vuota V1_list.
24          V1_list.append(V1)
25      return V1_list
```

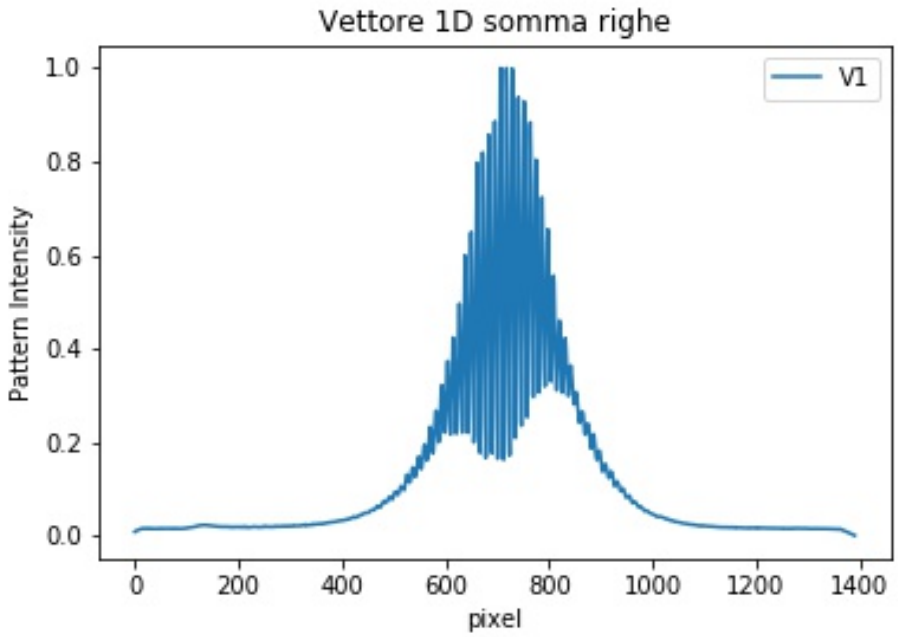

Figura 10: Grafico della figura di interferenza dell'immagine i-esima.

In figura 10 è mostrato il grafico di una figura di interferenza ottenuto con la funzione $V1$. Si passa poi a calcolare la derivata della figura di interferenza e

ad ottenerne il grafico.

Listing 5: libreria.py, funzione $Vder$.

```
# Input: lista dei valori dei vettori 1D.

# Output: lista dei valori dei vettori Vder.

def Vder(V1_list):
    DATA = []
    l = len(V1_list)
    for i in range(l):

        # Per ogni elemento della lista V1_list si calcola la derivata.
        Vder = np.diff(V1_list[i])
        Vder = np.array(Vder, dtype=np.float64)

        # Si crea e si salva il grafico della derivata della figura di
            interferenza.
        plt.plot(Vder, label = 'Vder')
        plt.xlabel('Pixel')
        plt.ylabel('Pattern Intensity')
        plt.title('Derivata vettore unidimensionale')
        plt.legend()
        plt.savefig('derivata_vettore_V_1D ' + str(i) + '.jpg')
        plt.close()

        # Si taglia il vettore Vder in un intervallo più piccolo.
        data = Vder[400:1000]

        # Si aggiunge l'array data alla lista vuota DATA e si creano dei
             file di testo con i valori di ogni array data.
        DATA.append(data)
    np.savetxt("DATA " + str(i)+ ".txt", data, header = 'data \n')
    return DATA
```

Di seguito il grafico della derivata della figura di interferenza.

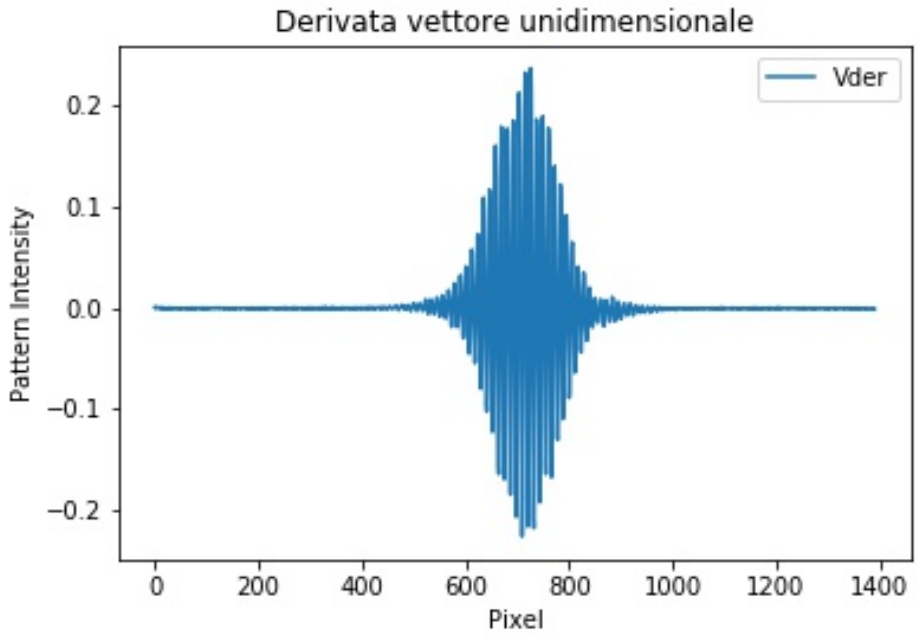

Figura 11: Grafico della derivata della figura di interferenza dell'immagine i-esima.

Si procede a calcolare il fit per ogni derivata della figura di interferenza ottenuta. Si definisce una funzione definita dai parametri che sono da ottimizzare, che rappresenterà la forma della curva del fit.

Listing 6: libreria.py, funzione *data fit*.

```
1 # Input: valori asse x, parametri che devono essere ottimizzati per
      ricavare la curva che interpola i dati.
2
3 # Output: equazione della curva che interpola i dati.
4
5 #si definisce una funzione, definita dai parametri che sono da
      ottimizzare
6 def data_fit(deltaD, est_sigma, est_k, est_phase, est_mean, est_D,est_amp
      ):
7     return est_amp*np.exp(-(((deltaD-est_D)/(est_sigma))**2)/2)*np.cos(
          est_k*deltaD+est_phase)+est_mean
```

La funzione *data fit* definisce la curva che meglio approssima i dati. Essa è descritta dall'equazione 29 ed è composta da una parte gaussiana e da una parte cosinusoidale. In *data fit* le due parti, quella gaussiana e quella cosinusoidale, sono state scritte nella loro forma generale. Il termine gaussiano $\exp\left(-\left(\frac{deltaD-est\ D}{est\ sigma}\right)^2/2\right)$ è descritto da: *deltaD* ovvero le ascisse, *estD* e *est sigma* si riferiscono rispettivamente alla media e alla deviazione standard (σ); questi due valori servono per stimare la posizione e la forma della distribuzione di Gauss. Il termine cosinusoidale è $\cos(est\ k \cdot deltaD + est\ phase)$ ed è descritto da: *est k* che corrisponde al parametro $\frac{4\pi}{\lambda}$ (da (29)) e *est phase*, la fase, che occorre per stimare lo spostamento orizzontale dei dati dalla posizione zero. Il termine *est amp* è il parametro che stimerà l'ampiezza della curva. Il termine *est mean* stima se la curva è localizzata più in alto o più in basso rispetto all'asse delle ascisse.

Il passo successivo è utilizzare *data fit* all'interno della funzione *fit*. Si trovano il fit e i valori ottimali estrapolati dal fit per ogni immagine.

Listing 7: libreria.py, funzione *data fit*.

```
1  # Input: lista dei valori dei vettori Vder
2
3  # Output: lista dei valori dei fit
4
5  def fit(DATA):
6      FIT = []
7      # Asse delle x da 0 a 600 punti, equivalente al numero di elementi di
           DATA[i]
8      N = 600
9      deltaD = np.linspace(0,599, N)
10     l = len(DATA)
11
12     # Si calcola il fit su ogni elemento della lista DATA.
13     for i in range(l):
14
15         # Si dichiarano dei valori approssimativi dei parametri della
               funzione data_fit da cui la funzione curve_fit inizia i
               calcoli.
16         p0 = [1, 0.6, 0, 0.0000, 320,0.2]
17
18         # La funzione curve_fit calcola il fit per mezzo dell' algoritmo
               'least squares'; minimizza la somma al quadrato della
               differenza tra i dati e una funzione definita da parametri (
               data_fit). La funzione curve_fit restituisce i parametri
               ottimali in unità di pixel e la matrice della covarianza dei
               parametri in unità di pixel.
19         popt, pcov = curve_fit(data_fit, deltaD, DATA[i], p0)
20
```

```
21          # Si crea e si salva il grafico del fit.
22          plt.plot(deltaD, DATA[i], label = 'data')
23          plt.plot(deltaD, data_fit(deltaD, *popt), label = 'fit')
24          plt.xlabel('pixel')
25          plt.ylabel('Pattern Intensity')
26          plt.title("fit "+str(i) )
27          plt.legend()
28          plt.savefig("fit "+str(i) + ".jpg")
29
30          #Si aggiunge alla lista FIT i valori del fit.
31          Fit = data_fit(deltaD, *popt)
32          FIT.append(Fit)
33          plt.close()
34
35      #Si ricavano gli errori dei parametri ottimali e si crea una lista.
36      COV = (np.sqrt(np.diag(pcov)))
37      cov = [COV[0], COV[1], COV[2],COV[3], COV[4], COV[5]]
38
39      # Si crea una lista dei parametri ottimali.
40      par_opt = [popt[0],popt[1],popt[2], popt[3], popt[4],popt[5]]
41
42      # Si salvano dei file di testo contenenti i valori dei parametri
            ottimali, dei fit e degli errori dei parametri.
43      np.savetxt("opt_par "+str(i)+".txt" ,par_opt , header = 'dall_alto in
             basso parametri in pixel \n est_sigma, est_k , est_phase ,
            est_mean , est_D , est_amp' )
44      np.savetxt("FIT "+ str(i)+ ".txt", data_fit(deltaD,*popt),header = '
            fit \n')
45      np.savetxt("errori parametri " + str(i)+ ".txt", cov, header = '
            errori \n est_sigma, est_k , est_phase , est_mean , est_D,
            est_amp')
46      return FIT
```

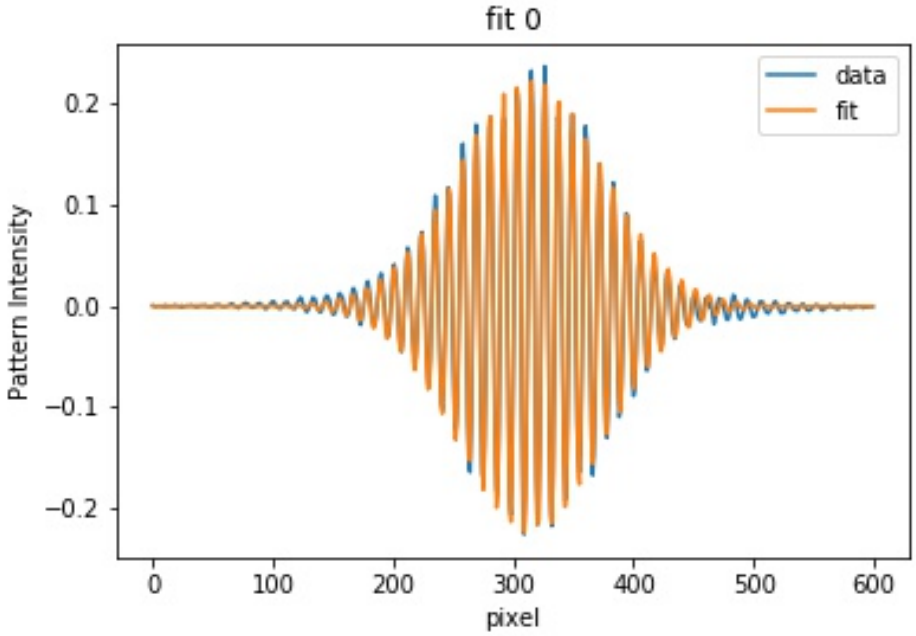

Figura 12: Grafico del fit sulla figura i-esima.

In figura 12 viene mostrato uno dei fit ottenuti.

Si è scritta la funzione *load valori* per ricavare una lista di valori relativi allo stesso parametro contenuto nei file di testo creati in precedenza. La funzione legge un file di testo contenente la lista dei nomi dei file da cui si vuole estrarre il valore. Questi file hanno la stessa struttura. *Load valori* è stata usata per ricavare la lista dei valori di sigma, del parametro k, del parametro D ovvero il centro della gaussiana e Phi la fase.

Listing 8: libreria.py, funzione *load valori*.

```
1 # Input: nome lista dei nomi dei file da cui si vuole estrarre il
      parametro, riga della lista del valore che si vuole leggere
2
3 # Output: lista dei valori dello stesso parametro.
4
5 def load_valori(lista1,e):
6     valori = []
```

```
    # Si apre in modo binario il file lista1
    with open(lista1, 'r') as c:
        # Si crea una lista leggendo gli oggetti contenuti in lista1
        lista_ = [line.strip() for line in c]
        l = len(lista_)

        # Per ogni oggetto di lista1 si estrae il valore relativo alla
            riga 'e' utilizzando la funzione load_valore descritta in
            precedenza. Si aggiungono i valori ricavati ad una lista.
        for i in range(l):
            with open(lista_[i], 'r') as f:
                valore = load_valore(f,e)
                valori.append(valore)
        return valori
```

Si definisce poi la funzione *media* per ricavare la media e la deviazione standard.

Listing 9: libreria.py, funzione *media*.

```
# Input: lista di valori

# Output: media, deviazione standard


def media(dati):
    # Si calcola la media con la funzione mean di numpy
    media = np.mean(dati)
    l = len(dati)

    # Si ricava la deviazione standard
    for i in range(l):
        delta = (dati[i]-media)**2
        arg = (np.sum(delta))/l
        strd = np.sqrt(arg)
    return media, strd
```

Per mezzo della funzione media si ricavano i valori medi dei parametri del fit: $\sigma = 60.68 \pm 0.03\ px$, $k = 0.55114 \pm 0.00009\ px$, $D = 321.08 \pm 0.03\ px$.

La funzione *find phi* permette di trovare la media dei valori della differenza tra il valore della fase di un fit di riferimento e i valori della fase degli altri fit.

Listing 10: libreria.py, funzione *find phi*.

```
# Input: lista dei valori delle derivate della figura di interferenza.

# Output: media dei valori della differenza tra il valore della fase di
    un fit di riferimento e i valori della fase degli altri fit.

```

```
5  def find_phi(DATA):
6      phase = []
7      l = len(DATA)
8      for i in range(l):
9
10         #Si esegue un fit all'interno di un intervallo di pixel piccolo
               per ricavare valori accurati della fase.
11         N = 20
12         deltaD = np.linspace(0,19, N)
13         dati = DATA[i]
14         data = dati[311:331]
15         p0 = [60, 0.6, 1, 0, 10, 0.2]
16         popt, pcov = curve_fit(data_fit, deltaD, data, p0)
17
18         # Si crea una lista con il valore delle fasi
19         phase.append(popt[2])
20
21     # Si trovano gli spostamenti tra il fit di riferimento (num.10) e gli
           altri fit.
22     f1 = []
23     f2 = []
24     for e in range(0,9):
25         F1 = abs(phase[10]-phase[e])
26         f1.append(F1)
27     for j in range(11,19):
28         F2 = abs(phase[10]-phase[j])
29         f2.append(F2)
30     p = f2+f1
31
32     # Si calcola la media e la deviazione standard dei valori di
           spostamento
33     fase, err_fase = media(p)
34     return fase, err_fase
```

Questo valore rappresenta il valore medio dello spostamento degli interferogrammi da un interferogramma di riferimento ed equivale a $\phi = 0.259 \pm 0.024\ px$. Servirà in seguito per calcolare l'incertezza sullo posizione relativa degli specchi calcolata con il valore della fase estrapolato dai fit.

Come descritto in precedenza la funzione *main* restituisce due valori per l'incertezza sulla posizione relativa degli specchi utilizzando due metodi: il primo utilizza il valore medio delle fasi ricavato dalla funzione *find phi*, il secondo consiste nell'ottenere il valore medio di spostamento per mezzo della correlazione. Per ottenere un valore più attendibile si esegue la correlazione con una risoluzione al di sotto del pixel, in particolare 0.01 pixel. Le prossime funzioni servono per questo scopo. La prima funzione definita è *highres* serve per ottenere una più alta risoluzione.

Listing 11: libreria.py, funzione *highres*.

```
# Input: il vettore che si vuole analizzare, l'inverso della risoluzione
    (res = 1 \ risoluzione), metodo di interpolazione.

# Output: ascisse con una risoluzione maggiore, ordinate in funzione di
    un intervallo dalla risoluzione maggiore

def highres(y, res,kind='cubic'):
    y = np.array(y)

   # Si crea un array x con lo stesso numero di elementi di y.
    x = np.arange(0,y.shape[0])

    # Si interpolano x e y.
    f = interp1d(x,y, kind='cubic')

    # Si definiscono delle nuove ascisse dalla risoluzione maggiore.
    xnew = np.linspace(0,x.shape[0]-1,x.shape[0]*res)

    # Si ottengono i valori delle y in funzione delle nuove coordinate x.
    ynew = f(xnew)
    return xnew, ynew
```

La funzione successiva ricava il valore di spostamento tra un fit (o i dati) e un fit (o i dati) di riferimento attraverso la correlazione con una risoluzione al di sotto del pixel.

Listing 12: libreria.py, funzione *cross corr sub pixel*.

```
# Input: array di riferimento, l'array di cui si vuole trovare lo
    spostamento, l'inverso della risoluzione, il numero di pixel da cui
    si ricava la regione di interesse in cui si esegue la correlazione.

# Output: valore di spostamento.

def cross_corr_sub_pixel(ref,target,res,delta_pix):
    l = len(ref)

    #Si definiscono i valori di start e stop dei pixel attorno alla
        coordinata del massimo, serviranno per creare la regione di
        interesse in cui agirà la correlazione
    start = np.argmax(ref)-delta_pix
    stop = np.argmax(ref)+(delta_pix)


```

```
13      # Si applica la funzione highres per ottenere una risoluzione
            maggiore
14      x,r1 = highres(ref[start:stop], res,kind='linear')
15      x,r2 = highres(target[start:stop],res,kind='linear')
16
17      # Si sottrae la media dei valori.
18      r1 -= r1.mean()
19      r2 -= r2.mean()
20
21      # Si esegue la correlazione tra i due array per mezzo della funzione
            correlate di numpy. Il metodo di correlazione scelto restituisce
            un array il cui numero di elementi è max(N,M),dove N e M sono le
            dimensioni dei due array su cui si esegue la correlazione.
22      c = np.correlate(r1,r2,'same')
23
24      # Si trova la coordinata relativa al massimo
25      p = np.array(np.argmax(c), dtype=np.float)
26
27      # Si trova il valore di spostamento riscalato per la risoluzione.
28      shift_value = (p-np.argmax(ref))*1/res
29      return shift_value
```

La funzione *find correlation* serve per applicare la funzione *cross corr sub pixel* tra il fit (dati) di riferimento e gli altri fit (dati) ottenuti in laboratorio.

Listing 13: libreria.py, funzione *find correlation*.

```
1  # Input: lista array su cui si vuole applicare la correlazione
2
3  # Output: lista dei valori di spostamento.
4
5  def find_correlation(lista):
6      l = len(lista)
7      C1 = []
8      C2 = []
9
10     # Si esegue la correlazione tra l'array di riferimento e gli altri
           array presenti nella lista. Si aggiungono i valori di spostamento
            ad una lista.
11     for i in range(0,9):
12         corr_fit = cross_corr_sub_pixel(lista[10],lista[i],100,3)
13         C1.append(abs(corr_fit))
14     for j in range(11,l):
15         corr_fit = cross_corr_sub_pixel(lista[10],lista[j],100,3)
16         C2.append(abs(corr_fit))
17     corr = C1+C2
18     return corr
```

In figura 13 viene mostrato lo spostamento tra due fit. Si può notare che i due massimi non cadono sullo stesso pixel.

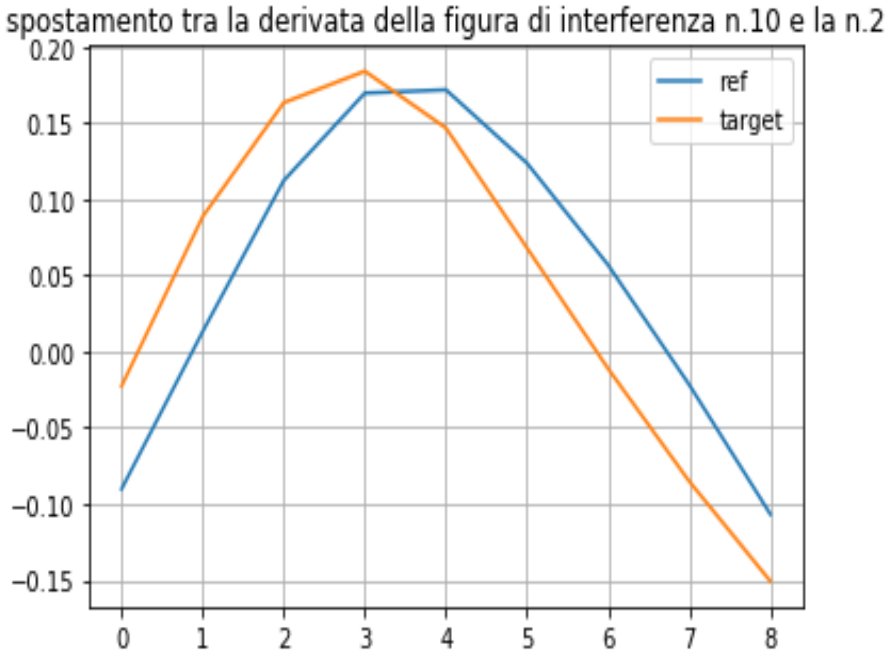

Figura 13: Grafico tra la derivata della figura di interferenza di riferimento (n.10) e la derivata della figura di interferenza di cui si vuole ottenere lo spostamento (n.2). Il grafico è ristretto ad un intervallo di 9 pixel.

Si applica poi la funzione media descritta in precedenza per calcolare il valore medio dei valori della lista corr. Il valore dello spostamento medio tra una derivata della figura di interferenza di riferimento e le altre derivate equivale a $\Delta x = 0.257 \pm 0.090\ px$.

Il passo successivo è ricavare il valore dell'angolo con cui sono inclinati tra loro gli specchi da (27), ovvero $\sin\alpha = \lambda/2x$. Il parametro λ è il valore della lunghezza d'onda centrale del SLED, mentre x, la distanza tra due massimi successivi, si ricava da $x = 4\pi/k$, dove k è il parametro che viene estrapolato dai fit. Il valore x è in unità di pixel perciò occorre portarlo in unità metriche decimali. Inoltre bisogna tenere in considerazione che nello strumento costruito in laboratorio è presente una lente che focalizza i fronti d'onda sul sensore, perciò bisogna

dividere x per il valore della magnificazione. La seguente funzione *angle tilt* esegue questi passaggi.

Listing 14: libreria.py, funzione *angle tilt*.

```
# Input: il parametro k, magnificazione, dimensione del pixel, lunghezza
    d'onda centrale SLED.

# Output: angolo di tilt degli specchi.

def angle_tilt(k, mag, px_size, L_0):
    I = px_size/mag

    # Si calcola la distanza tra i massimi successivi prima in unità di
        pixel poi in metri.
    x_px = (4*np.pi)/k
    x = x_px*I

    # Si trova l'angolo di tilt
    senalfa = L_0/(2*x)
    alfa = (np.arcsin(senalfa)*180/(np.pi))
    return alfa
```

Si calcola poi l'errore dell' angolo di tilt, utilizzando le formule della propagazione dell'errore.

Listing 15: libreria.py, funzione *delta angle tilt*.

```
# Input: il parametro k, l'errore del parametro k, magnificazione,
    dimensione del pixel, lunghezza d'onda centrale SLED, errore della
    lunghezza d'onda centrale SLED.

# Output: errore dell'angolo di tilt.

def delta_angle_tilt(k, deltak, mag, px_size, L_0, deltaL_0):
    I = px_size/mag
    x_px = (4*np.pi)/k
    x = x_px*I

    # Si trova l'errore della distanza tra i massimi successivi prima in
        pixel poi in metri
    deltax_px = ((4*np.pi)/(k**2))*deltak
    deltax = deltax_px*I

    # Si calcola l'errore dell'angolo di tilt applicando la formula di
        propagazione dell'errore.
```

```
15      senalfa = L_0/(2*x)
16      t1 = L_0/((2)*(x)**2)
17      t2 = 1/(2*x)
18      DD = 1/np.sqrt(1-(senalfa)**2)
19      delta_angle_tilt = (abs(DD*t1)*deltax+abs(DD*t2)*deltaL_0)*180/(np.pi
            )
20      return delta_angle_tilt
```

Per l'angolo di tilt si ottiene $\alpha = 0.06149 \pm 0.00008$ °.

Si esegue poi la funzione *accuratezza*. Conoscendo il valore dell'angolo di tilt e lo spostamento medio delle immagini si ottengono i valori dell'incertezza sulla posizione relativa degli specchi dell'interferometro.

Listing 16: libreria.py, funzione *accuratezza*.

```
1  # Input: angolo di tilt, valore medio spostamento (correlazione o fase),
       dimensione pixel, magnificazione.
2
3  #Output: incertezza sulla posizione relativa degli specchi dell'
       interferometro.
4
5  def accuratezza(alfa,shift, px_size, mag):
6      I = px_size/mag
7
8      # Incertezza sulla posizione zero dello specchio.
9      deltaD = (np.tan(np.deg2rad(alfa)))*shift*I
10     return deltaD
```

La seguente funzione ha il compito di calcolare l'errore dell'incertezza.

Listing 17: libreria.py, funzione *delta accuratezza*.

```
1  # Input: angolo di tilt, errore angolo di tilt, valore di spostamento
       medio, errore valore di spostamento, dimensione del pixel,
       ingrandimento.
2
3  #Output: errore dell'incertezza sulla posizione relativa degli specchi.
4
5  def delta_accuratezza(alfa, delta_alfa, shift, delta_shift, px_size, mag)
       :
6      I = px_size/mag
7      Shift = shift*I
8      Delta_shift = delta_shift*I
9
10     # Si calcola l'errore per mezzo della formula di propagazione degli
           errori.
11     errore = (abs(Shift/(np.cos(np.deg2rad(alfa)**2)))*delta_alfa + abs((
           np.tan(np.deg2rad(alfa))))*Delta_shift)
12     return errore
```

5.2 La procedura find angle

La procedura *find angle* serve per trovare l'angolo di inclinazione delle frange che viene impiegato per ruotare le immagini all'interno della procedura *main*. La struttura di *find angle* è mostrata nello schema seguente:

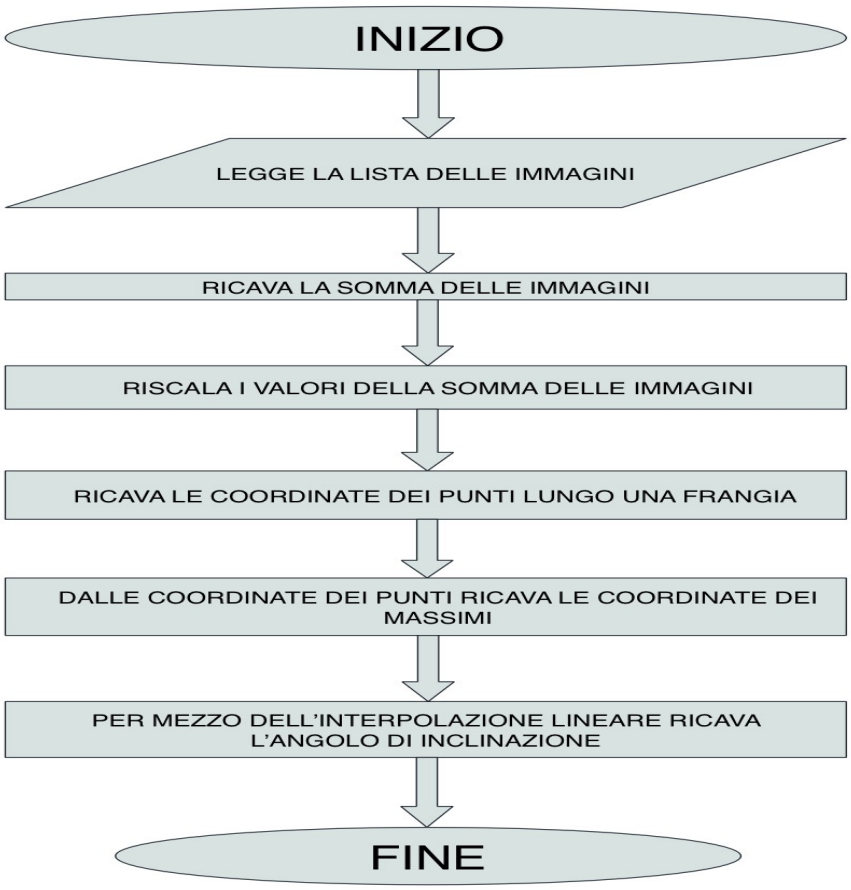

Figura 14: Schema della funzione find angle. La funzione restituisce l'angolo di inclinazione delle frange.

Si mostra di seguito il codice sorgente della funzione *find angle*.

Listing 18: main.py, *find angle*.

```
# Input: file di testo contenente la lista dei file binari

# Output: angolo di inclinazione delle frange.

def find_angle(r):
    # legge e somma tra loro le venti immagini.
    somma_array = apertura(r)

    # Riscala i valori della somma delle immagini.
    imaOut = tvscl(somma_array)

    # Chiama l'evento mouse per selezionare le coordinate.
    x1,y1 = call_on_mouse(imaOut, 'somma_immagini')

    # Si trovano le x dei massimi lungo una frangia.
    maxX = get_xMax(x1,y1,imaOut)

    # Tramite l'interpolazione lineare si ricava il valore dell'angolo.
    angle = interpolazione_lineare(maxX,y1)

    return angle
```

La funzione *apertura* recupera da un file di testo i nomi dei file contenuti e poi li somma tra loro per ottenere una migliore qualità dei dati.

Listing 19: libreria.py, *apertura*.

```
# Input: nome del file di testo contenente la lista dei file binari

# Output: immagine della somma tutti i file binari.

def apertura(r):
    # Si apre in modo binario il file di testo contenente la lista dei
        file binari
    with open(r, 'rb') as c:

        # Si crea una lista leggendo gli oggetti contenuti in r.
        lista_immagini = [line.strip() for line in c]

        l = len(lista_immagini)

        # Si dichiara una matrice di zeri unsignedinteger64 con
            dimensioni pari a quelle del sensore.
        somma_array = np.zeros((1040,1392),dtype=np.uint64)

```

```
17          # Si dichiara una matrice 1D di zeri in formato unsignedinteger64
                con un numero di elementi pari a (1040 X 1392).
18          somma = np.zeros((1040*1392),dtype=np.uint64)
19
20          # Ad ogni iterazione del ciclo for si apre un file binario.
21          for i in range(l):
22              with open(lista_immagini[i], 'rb') as f:
23                  data = np.fromfile(f,dtype=np.uint16)
24
25                  # Si inverte l'ordine dei bytes.
26                  swap = data.byteswap()
27
28                  # Ad ogni iterazione si aggiungono le immagini alla
                        matrice vuota somma.
29                  somma=somma+np.array(swap,dtype=np.uint64)
30
31          # Si ottiene l'immagine con le dimensioni del sensore.
32          somma_array = np.reshape(somma, (1040,1392))
33      return somma_array
```

L'immagine che si ottiene è mostrata nella figura seguente.

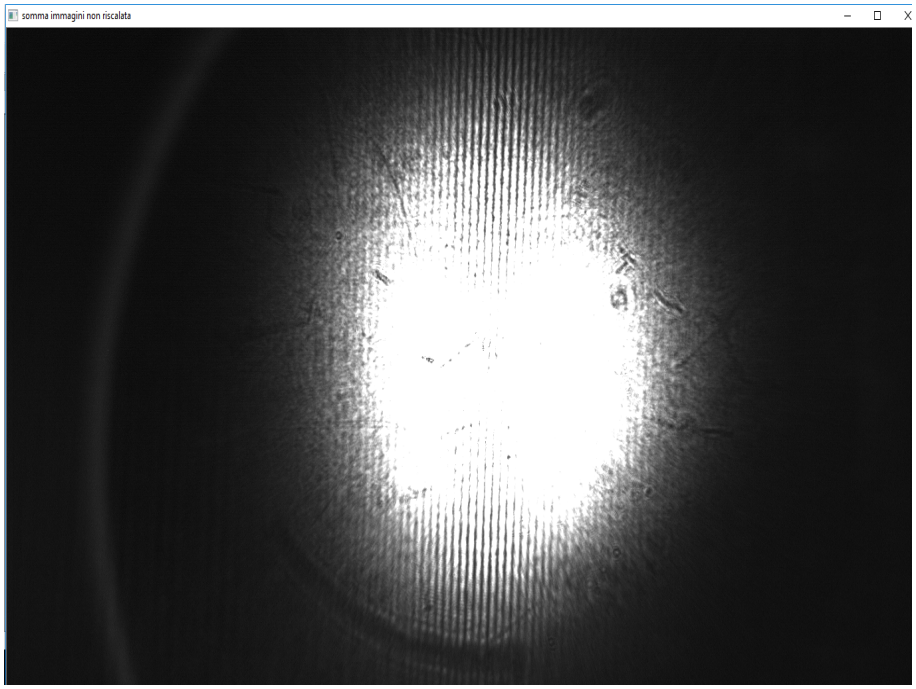

Figura 15: Somma delle immagini non riscalata.

Come si vede in figura 15 i valori dell'immagine sono saturi. La seguente funzione *tvscl* ha il compito di riscalare i valori dell'immagine.

Listing 20: libreria.py, *tvscl*.

```
# Input: somma immagini non riscalata

# Output: somma immagini riscalata

def tvscl(imaIn):
  # Si determinano il massimo e il minimo dell'immagine.
  ma=np.amax(imaIn)
  mi=np.amin(imaIn)

  # Si cambia il formato in float.
  imaInf=np.array(imaIn,dtype=np.float)

  # Si riscalano i valori dell'immagine.
  imaOut=((imaInf-mi)/(ma-mi))
  return imaOut
```

Come si osserva dalle figure 6 e 15 le frange sono inclinate di un certo angolo. Per trovare questo angolo si ricercano le coordinate dei massimi lungo la stessa frangia ma a causa delle imperfezioni sugli specchi le posizioni dei massimi non riproducono esattamente una retta. Questa retta viene trovata interpolando linearmente le diverse posizioni dei massimi lungo la frangia. Viene poi calcolato l'angolo tra la retta e l'asse x del grafico, che corrisponde all'angolo di inclinazione delle frange. Le prossime funzioni descritte servono a questo. La prima funzione esposta, *call on mouse*, serve per selezionare sull'immagine, tramite i click del mouse, le coordinate di alcuni punti lungo la stessa frangia.

Listing 21: libreria.py, *call on mouse*.

```
#Input: somma immagini riscalata, nome finestra di visualizzazione.


#Output: lista delle coordinate x dei punti, lista delle coordinate y dei
     punti.

def call_on_mouse(imaOut, nome_finestra):
    #Si dichiarano due liste vuote .
    x1 = []
    y1 = []

    #Si definisce una funzione on_mouse per chiamare l'evento mouse sull'
        immagine.
```

```
12      def on_mouse(event,x,y,flags,param):
13
14          # L'evento mouse corrisponde al click del tasto sinistro del
                mouse
15          if event == cv2.EVENT_LBUTTONDOWN:
16
17              # Ad ogni click si aggiungono alle liste x1 e y1 le coordinate
                    ottenute dai click.
18              x1.append(x)
19              y1.append(y)
20          return x1,y1
21
22      # Si apre l'immagine
23      cv2.imshow(nome_finestra, imaOut)
24
25      # Si chiama l'evento mouse sull'immagine, si dichiarano le due liste
            per fare in modo che tutte le posizioni dei click vengano
            memorizzate.
26      cv2.setMouseCallback(nome_finestra,on_mouse, (x1,y1))
27
28      # Si imposta il tempo con cui verrà chiusa la finestra, in questo
            caso infinito; perciò la finestra si chiude usando l'apposita
            icona.
29      cv2.waitKey(0)
30      cv2.destroyAllWindows
31      return x1,y1
```

Di seguito un esempio della finestra in cui si utilizza la funzione *call on mouse*.

Figura 16: Somma delle immagini riscalata sulla quale si selezionano dei punti lungo una frangia. Come esempio vengono mostrati in giallo i possibili punti selezionati. Viene inoltre mostrata l'orientazione degli assi x e y dell'immagine.

La successiva funzione utilizza le coordinate, trovate in precedenza, per determinare le corrette ascisse dei diversi massimi.

Listing 22: libreria.py, funzione *get x max*

```
#Input: coordinate x dei click, coordinate y dei click, somma immagini
    riscalata.

#Output: lista delle coordinate x dei massimi lungo una frangia.

def get_xMax(x1,y1,imaOut):
    maxX = []
    l = len(x1)

    # Ad ogni iterazione del ciclo si selezionano le coordinate x e y dei
        punti.
    for i in range(l):
        nriga1 = np.array(y1[i], dtype = np.uint32)
        ncolonna1 = np.array(x1[i], dtype= np.uint32)
```

```
13
14          # Si definisce il dataframe di pandas, ImaOut
15          array_df = pd.DataFrame(imaOut)
16
17          # Si converte il valore in uno scalare poiché (iloc) la funzione
                seguente prende in considerazione solo scalari.
18          a = np.asscalar(np.array(nriga1))
19
20          # Si seleziona dall'immagine la riga corrispondente alla
                coordinata y.
21          riga1 = array_df.iloc[a]
22
23          # Si taglia la riga in un intervallo di 8 px attorno al valore
                della coordinata x.
24          X1 = np.asscalar(np.array(ncolonna1))
25          rigacorta1 = riga1[(X1-4):(X1+4)]
26
27          #Si crea e si salva il grafico del massimo.
28          r1 = pd.DataFrame(rigacorta1)
29          l1 = plt.plot(r1)
30          plt.xlabel('pixel')
31          plt.ylabel('intensity')
32          plt.title('massimo')
33          plt.savefig('Max ' + str(i) +'.jpg')
34
35          # Si ottengono i valori delle coordinate x del grafico.
36          xarray1 = l1[0].get_xdata((l1))
37          xarray1 = np.array(xarray1, dtype = np.float32)
38
39          # Si ottengono i valori delle coordinate y del grafico.
40          yarray1 = l1[0].get_ydata((l1))
41          yarray1 = np.array(yarray1, dtype = np.float32)
42
43          # Si ricava la coordinata x relativa al massimo.
44          Max1 = xarray1[yarray1.argmax()]
45
46          # Ad ogni iterazione del ciclo si aggiunge alla lista vuota maxX
                il valore Max1.
47          maxX.append(Max1)
48          plt.close()
49      return maxX
```

Si mostra il grafico, ottenuto con *get x max*, di uno dei massimi.

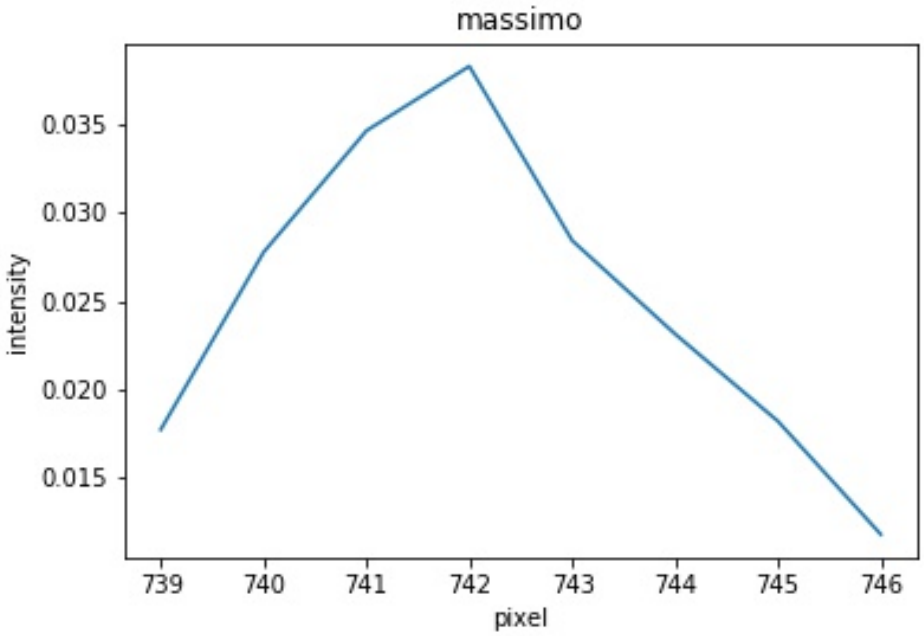

Figura 17: Grafico di un massimo nell'intervallo di valori attorno ad un punto i-esimo selezionato sull'immagine. La funzione *get x max* restituisce una lista dei valori delle ascisse dei massimi.

Ottenute le coordinate dei massimi si può calcolare l'angolo di inclinazione delle frange per mezzo dell'interpolazione lineare. La funzione *interpolazione lineare* esegue questo passaggio. Una parte di questa funzione è stata tratta da "*Dispense del corso di Sperimentazioni di Fisica* 1 *modulo A*" [Baruffolo, 2016]. Per calcolare la retta che meglio approssima i dati si sono utilizzate le seguenti equazioni. Se la retta è del tipo $y = C + mx$, i coefficienti della retta e i corrispettivi errori si calcolano con le formule seguenti:

$$\Delta = N\sum x^2 - \left(\sum x\right)^2 \tag{30}$$

$$C = \frac{\sum x^2 \sum y - \sum x \sum xy}{\Delta} \tag{31}$$

$$m = \frac{N\sum xy - \sum x \sum y}{\Delta} \tag{32}$$

$$\sigma_y = \sqrt{\frac{1}{N-2}\sum_{i=1}^{N}(y_i - C - mx_i)^2} \tag{33}$$

$$\sigma_C = \sigma_y\sqrt{\frac{\sum x^2}{\Delta}} \tag{34}$$

$$\sigma_m = \sigma_y\sqrt{\frac{N}{\Delta}} \tag{35}$$

Listing 23: libreria.py, funzione *interpolazione lineare*.

```
#Input: lista delle coordinate x dei massimi, lista delle coordinate y
    dei massimi

#Output: angolo di inclinazione delle frange

def interpolazione_lineare(maxX,y1):
# y = C + m*x

    #Si definiscono le ascisse e le ordinate.
    maxX = np.asarray(maxX, dtype=np.float)
    y1 = np.asarray(y1, dtype=np.float)
    x = y1
    y = maxX

    # Parametri per calcolare le variabili seguenti.
    N = len(x)
    sumx = np.sum(x)
    sumy = np.sum(y)
    sumx2 = np.sum(x*x)
    sumxy = np.sum(x*y)

    # Rispettivamente eq. 30, eq.31, eq.32.
    delta = N*sumx2 - sumx*sumx
    C = (sumx2*sumy - sumx*sumxy)/delta
    m = (N*sumxy - sumx*sumy)/delta

    # Valore per calcolare le variabili seguenti.
    diffs = y-C-m*x

    # Rispettivamente eq. 33, eq.34, eq.35.
    sigy = np.sqrt(np.sum(diffs*diffs)/(N-2))
    sigC = sigy*np.sqrt(sumx2/delta)
    sigm = sigy*np.sqrt(N/delta)

    #Si definisce la retta con i valori appena ottenuti.
```

```
35      retta = C+m*x
36
37      #Si crea e si salva il grafico dell'interpolazione lineare.
38      plt.plot(x,y,'.', label = 'misure' )
39      plt.plot(x,retta,label = 'interpolazione')
40      plt.gca().invert_yaxis()
41      plt.xlabel('pixel')
42      plt.ylabel('pixel')
43      plt.title('interpolazione lineare')
44      plt.legend()
45      plt.savefig('interpolazione lineare.jpg')
46      plt.close()
47
48      #Si ricava l'angolo di inclinazione e il suo errore.
49      angle = abs(np.arctan(m)*180/(np.pi))
50      sig_angle = abs(1/(1+m**2))*sigm
51      dati = [angle, sig_angle, C, sigC, sigy]
52      np.savetxt('interpolazione lineare.txt', dati, header = 'angle,
            sig_angle, C, sigC, sigy \n')
53      return angle
```

Si evidenzia che per ottenere l'angolo di inclinazione cercato si sono ridefinite le ascisse con la lista delle coordinate y dei massimi e le ordinate con la lista delle coordinate x dei massimi. Si crea il grafico (invertendo il verso dell'asse y). Si trova l'angolo tramite l'arcotangente del coefficiente angolare m. L'errore associato all'angolo si calcola con la formula della propagazione degli errori: $\sigma_{angle} = \left|\frac{\partial angle}{\partial m}\right| \sigma_m$. Si salva su disco un file *interpolazione lineare.txt* contenente tutti i valori calcolati dall'interpolazione. Il grafico della retta che meglio approssima i dati è il seguente:

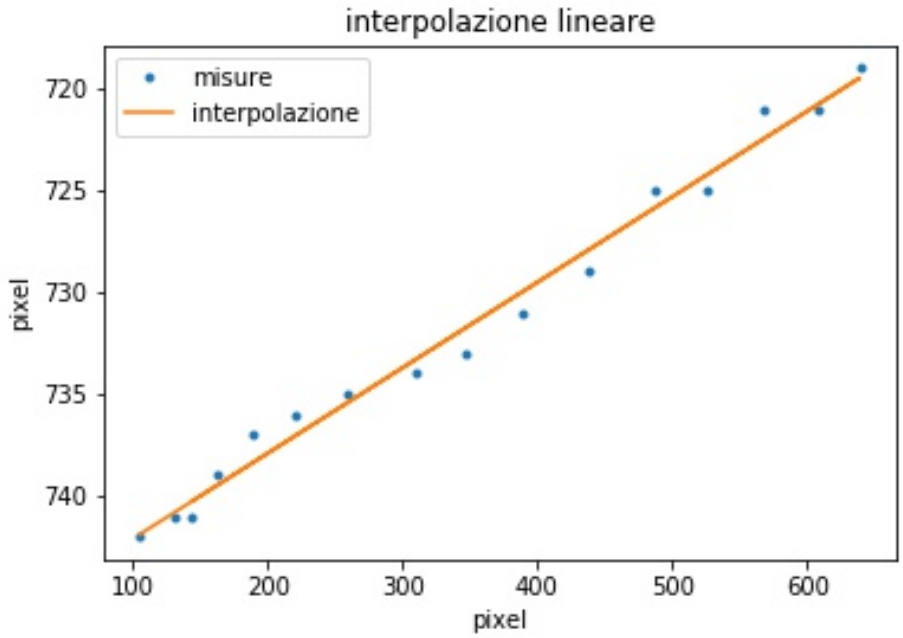

Figura 18: Grafico dell'interpolazione lineare tramite cui si ottiene l'angolo di inclinazione delle frange.

Il valore dell'angolo di inclinazione delle frange, ricavato tramite la funzione $find\ angle$, risulta essere $\theta = 2.405 \pm 0.001°$.

6 Conclusioni

L'interferometro a bassa coerenza può essere usato come misuratore di distanza, range finder. Questo strumento permette di calcolare la distanza fra due elementi ottici, parametro importante per un allineamento corretto. Alcuni esempi sono l'allineamento fra i due specchi, primario e secondario, di un telescopio, l'allineamento delle ottiche di una camera e gli esempi riportati nel paragrafo 4.1.1 ovvero l'utilizzo nell'ambito della missione PLATO e l'utilizzo per la calibrazione dello strumento MICADO per E-ELT.

In questo lavoro si cerca di stabilire quale sia l'accuratezza nella misura di distanza utilizzando un interferometro a bassa coerenza in condizioni realistiche. L'accuratezza è limitata da diversi fattori esterni quali deformazioni termomeccaniche, come ad esempio il cambio della temperatura durante il giorno dell'ambiente di misura, e possibili effetti della turbolenza atmosferica.

Dopo aver costruito in laboratorio un interferometro a bassa coerenza in configurazione di Michelson e catturato 20 immagini, una al secondo, prodotte dall'interferometro sul rivelatore, si è scritta una procedura in linguaggio Python per l'analisi dei dati. Con questa procedura sono state estratte, come primo passo, le figure di interferenza generate dall'interferometro. Sono state ricavate poi le derivate della figura di interferenza in modo da isolare il segnale della luminosità di fondo e ottenere così una curva più semplice da analizzare. Su questa curva sono stati calcolati poi i fit e sono stati ricavati i parametri dell'interferogramma. La curva di fit ottimale è una sovrapposizione di una componente sinusoidale e di una gaussiana tipica della sorgente a bassa coerenza.

Lo scopo principale della procedura è ricavare il valore dell'incertezza sulla posizione relativa degli specchi nei due bracci dell'interferometro, ovvero della differenza fra le loro lunghezze. Questa incertezza viene calcolata in due modi differenti.

Entrambi i metodi ricercano un valore di spostamento medio tra una curva interferometrica, scelta come riferimento, e le altre 19 ottenute in fase di acquisizione. Ciò che cambia tra i due procedimenti sono le curve interferometriche che vengono analizzate e il modo con cui viene ricavato il valore medio dello spostamento.

Il primo metodo ricava il valore dello spostamento medio eseguendo una correlazione diretta tra i dati degli interferogrammi.

Il secondo metodo, a differenza del primo, calcola la differenza di fase dai dati del fit. Il valore di spostamento medio si ricava tramite la differenza tra il valore della fase di un fit di riferimento e i valori delle fasi degli altri fit. Si ottengono quindi due valori per l'incertezza sulla posizione zero dello specchio.

Con il primo metodo, quello che utilizza la correlazione, il valore trovato per l'incertezza sulla posizione zero dello specchio è:

$$\Delta D = 4.6 \pm 1.9 \ nm.$$

Con il secondo metodo, quello che utilizza il valore delle fasi ottenute dal fit, si ricava un valore per l'incertezza pari a:

$$\Delta D = 4.6 \pm 0.8 \ nm.$$

Questi due valori trovati si riferiscono al nostro strumento costruito in laboratorio ma i due metodi esposti per ricavare la stima dell'accuratezza dello strumento possono essere utili per trovare i valori dell'accuratezza relativi a diverse applicazioni in cui si può utilizzare un interferometro a bassa coerenza.

Riferimenti bibliografici

[Baruffolo, 2016] Baruffolo, A. (2016). Dispense del Corso di Sperimentazioni di Fisica 1, modulo A.

[Chinellato et al., 2010] Chinellato, S., Pernechele, C., Carmignato, S., and Manzan, F. (2010). Surface measurements of radio antenna panels with white-light interferometry. *Proceedings of SPIE-The International Society for Optical Engineering*.

[Courteville et al., 2004] Courteville, A., Garcia, F., and Nel, L. (2004). Positioning of Optical Payload – SALT Telescope. *Proceedings of SPIE-The International Society for Optical Engineering*.

[Courteville et al., 2005] Courteville, A., Wilhelm, R., Delaveau, M., and Garcia, F. (2005). Non-contact in-process metrology using a high-accuracy low-coherence interferometer.

[D'Onofrio, 2004] D'Onofrio, M. (2004). Elementi di Ottica per Astronomi. Lezioni del corso di Laboratorio di Astronomia 1.

[D'Onofrio, 2014] D'Onofrio, M. (2014). Dispense del Corso di Ottica Applicata. Slides.

[Mazzoldi et al., 2012] Mazzoldi, P., Nigro, M., and Voci, C. (2012). *Elementi di Fisica: Elettromagnetismo e Onde*. EdiSeS.

[Pernechele and Chinellato,] Pernechele, C. and Chinellato, S. 3D surface measurements with Low Coherence Interferometry.

[Pernechele et al., 2009] Pernechele, C., Chinellato, S., Manzan, F., and Galeotti, M. (2009). Multi-thread real- time data processing for an image-based partially coherent light interferometer. *EURASIP*.

[Pernechele et al., 2017a] Pernechele, C., Fantinel, D., Magrin, D., Lessio, L., and Rodeghiero, G. (2017a). Low coherence interferometry-based meter-distance range finder. *IEEE*.

[Pernechele et al., 2017b] Pernechele, C., Magrin, D., and Rodeghiero, G. (2017b). Optical plate thickness measurement with single-shot low coherence interferometry. *OSA (Optical Society of America, 2017) Technical Digest (online)*.

[Rodeghiero et al., 2018] Rodeghiero, G., Pott, J.-U., Münch, N., Rohloff, R.-R., Grözinger, U., Biancalani, E., Sawczuck, M., Häberle, M., Moreno-Ventas, J., Schäfer, S., Seemann, U., Naranjo, V., Barboza, S., Müller, F., Hofferbert, R., Ramos, J., Mohr, L., Cárdenas Vázquez, M., Bizenberger, P., Pernechele, C., Ebert, M., and Fabricius, M. (2018). The MICADO first light imager for the ELT: preliminary design of the MICADO Calibration Assembly. *Proceedings of SPIE-The International Society for Optical Engineering*.

[Villa et al., 2010] Villa, F., D'Arcangelo, O., Pecora, M., Figini, L., Nesti, R., Simonetto, A., Sozzi, C., Sandri, M., Battaglia, P., Guzzi, P., Bersanelli, M., Butler, R., and Mandolesi, N. (2010). Planck LFI flight model feed horns. *Journal of Instrumentation*.