Co-funded by the
Erasmus+ Programme
of the European Union

PROGETTO HOLOMAKERS

Incentivare gli studenti delle scuole superiori a intraprendere un percorso nelle discipline STEM attraverso la creazione di ologrammi e altri processi innovativi per la creazione di immagini virtuali in diretta connessione con le ricerche moderne e la pratica in laboratorio

Erasmus+ KA2 2017-1-PL01-KA201-038420

Output 1

HOLOMAKERS Guida tecnica di riferimento

Partner capofila: WUT

Autore: Artur Sobczyk (WUT)

Diffusione: *Pubblica*

Versione: *01*

Stadio: *Finale*

Data: *2018*

Co-funded by the
Erasmus+ Programme
of the European Union

Contributi

. Karol Kakarenko, Warsaw University of Technology
. Rene Alimisi, EDUMOTIVA

Dichiarazione

Questo report è stato redatto nel contesto del progetto HOLOMAKERS. Laddove siano stati utilizzati altri materiali pubblicati e non, è stato segnalato.

Copyright

Clausola di finanziamento

Questo progetto è stato finanziato con il supporto della Commissione Europea. Il sostegno della Commissione europea alla produzione di questa pubblicazione non costituisce un'approvazione del contenuto, che riflette esclusivamente il punto di vista degli autori, e la Commissione non può essere ritenuta responsabile per l'uso che può essere fatto delle informazioni ivi contenute.

Co-funded by the
Erasmus+ Programme
of the European Union

Co-funded by the
Erasmus+ Programme
of the European Union

1. Imparare a conoscere le onde

1.1. Le onde in fisica

In fisica, l'onda è una perturbazione che si propaga attraverso un mezzo o nello spazio. Ad esempio, se si tira un sasso nell'acqua, si produrrà un'onda sulla superficie, propagandosi più lontano rispetto al punto in cui il sasso è caduto. In questo caso, la forza d'impatto si convertirà in oscillazioni del mezzo di propagazione (l'acqua). Un altro esempio è l'onda sonora. Anche in questo caso, l'onda viaggia grazie alle oscillazioni del mezzo di propagazione (l'aria).

Possiamo chiamare onda anche un'oscillazione limitata nello spazio. Ad esempio, far muovere un pendolo fa sì che il moto oscillatorio ritorni ogni volta nello stesso punto. In questo caso, l'onda coinciderà con l'oscillazione del pendolo che cambia nel tempo. La Fig. 1 illustra il cambiamento dell'oscillazione del pendolo nel tempo. Il pendolo devia nell'intervallo [-A, A], quindi possiamo dire che A coincide con l'ampiezza dell'oscillazione.

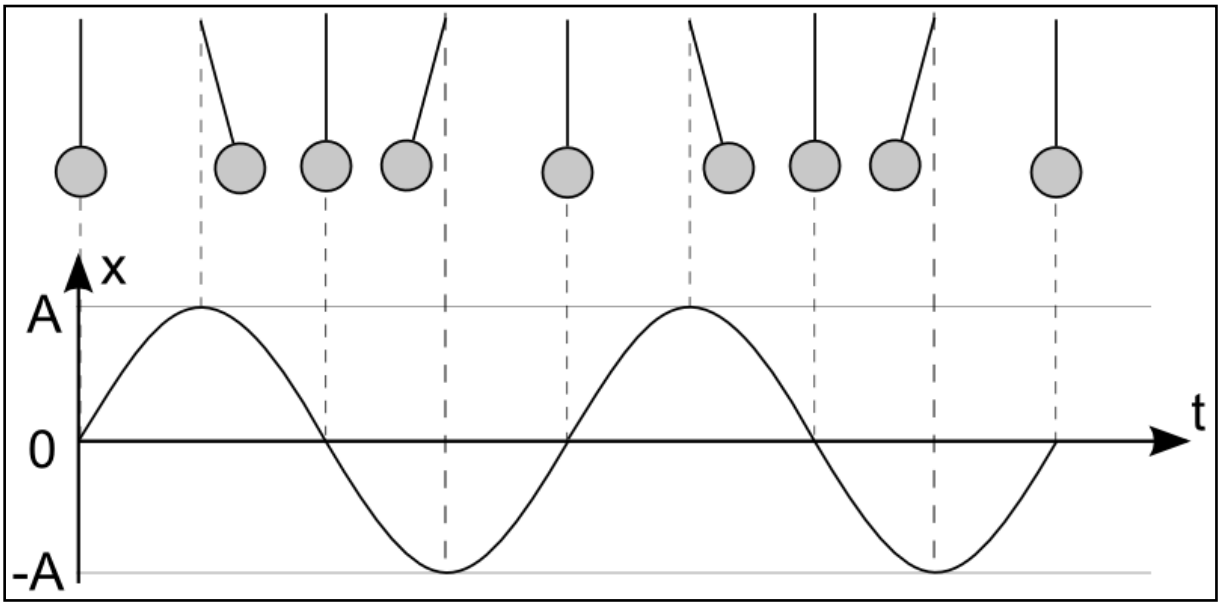

Fig. 1 Spostamento del pendolo in momenti diversi

Molto spesso le oscillazioni vengono descritte utilizzando la funzione seno. Ciò accade nell'esempio del pendolo appena menzionato, ma anche in quello del peso attaccato a una molla, o della corda di uno strumento musicale. Si tratta del tipo basilare di oscillazione. Questo genere di oscillazioni si chiamano oscillazioni **armoniche**. Come vedremo più avanti, ogni altro più complesso tipo di oscillazione consiste in un'oscillazione armonica. In conclusione, l'onda armonica si può definire tramite la funzione seno. Nel prossimo capitolo vedremo le proprietà essenziali delle onde.

1.2. Caratteristiche descrittive di un'onda

1.2.1. Ampiezza

L'ampiezza dell'onda è la distanza tra il punto di massima flessione e la posizione di equilibrio. Nel caso del pendolo, l'ampiezza dell'oscillazione è la sua flessione massima (vedi Fig. 2). Il valore massimo di un'onda si chiama cresta e il valore minimo, ventre.

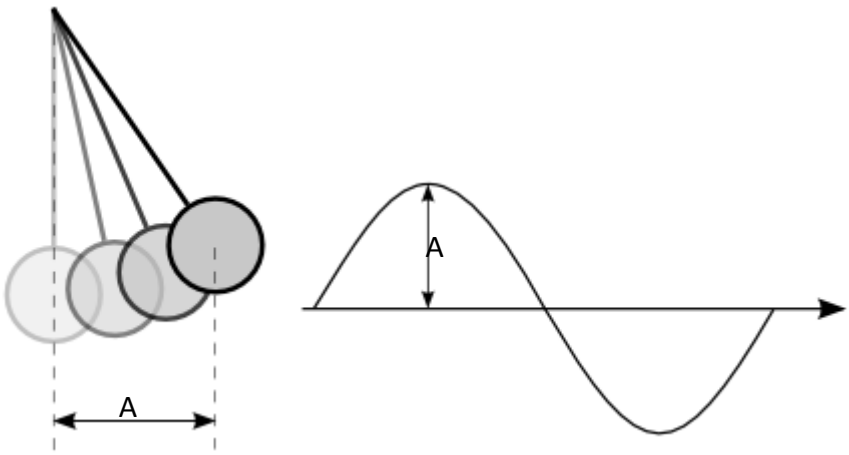

Fig. 2 Ampiezza delle oscillazioni di un pendolo

1.2.2. Frequenza

La frequenza ci dice quante sono le oscillazioni per unità di tempo (in un secondo). L'unità di frequenza si misura in Hz (Hertz). Ad esempio, 10Hz corrispondono a 10 oscillazioni al secondo, 15.5Hz a 15.5 oscillazioni al secondo. La Fig. 3 mostra due onde con differenti frequenze. La linea rossa è un'oscillazione completa. La frequenza della prima onda è di 3 Hz (3 oscillazioni al secondo) e la seconda di 6 Hz (6 oscillazioni al secondo).

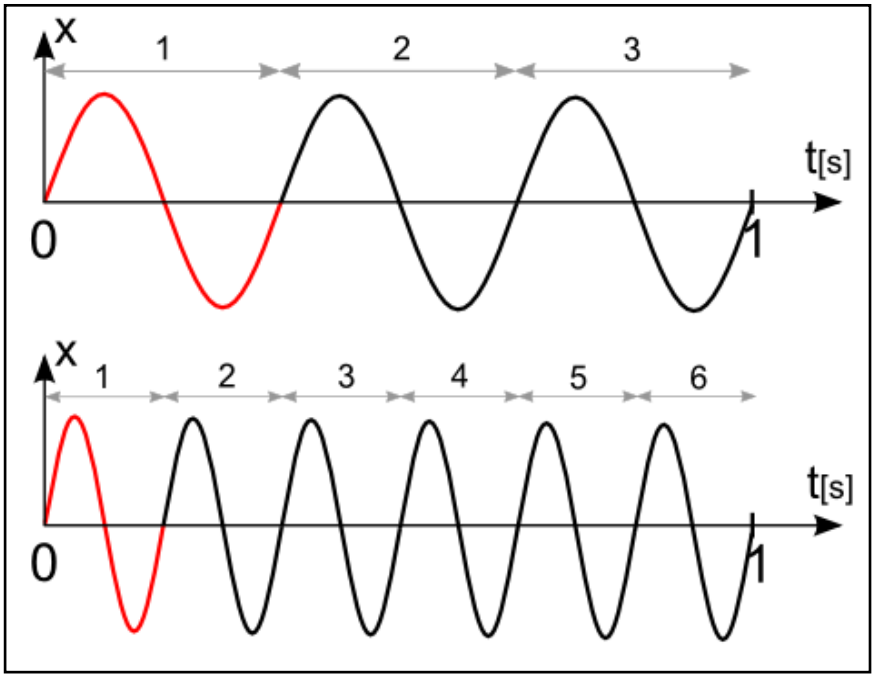

Fig. 3 Frequenza delle onde

1.2.3. Periodo

Il periodo dell'onda è direttamente connesso alla frequenza. Corrisponde al tempo (misurato in secondi) in cui l'onda compie un'oscillazione completa (segnata in rosso nella Fig. 3). Il periodo dell'onda (T) è connesso alla sua frequenza (*f*) in un rapporto di dipendenza $T = \frac{1}{f}$. Quindi, il periodo della prima onda nella Fig. 3 è di $\frac{1}{3}s$ mentre il periodo della seconda è uguale a $\frac{1}{6}s$

1.2.4. Lunghezza d'onda

Se l'onda si propaga nello spazio è possibile determinarne la lunghezza. Essa corrisponde alla distanza che l'onda raggiunge in un determinato periodo di tempo. La lunghezza d'onda (λ) è data dall'equazione $\lambda = vT$, dove v è la velocità di propagazione dell'onda e T il periodo.

1.2.5. Fase

La fase determina in quale istante del periodo di sviluppo dell'onda è collocato un dato punto. In pratica, tuttavia, non è tanto importante la singola fase dell'onda, quanto la differenza di fase tra onde. In altre parole, è semplicemente uno spostamento da un'onda all'atra. La Fig. 4a mostra una situazione in cui non c'è differenza di fase, mentre la Fig. 4b mostra due onde con un certo sfasamento. La differenza di fase può essere misurata tra due punti corrispondenti qualunque. Ad esempio, nella Fig. 4, per comodità, lo sfasamento è illustrato come la distanza tra i punti più alti delle onde.

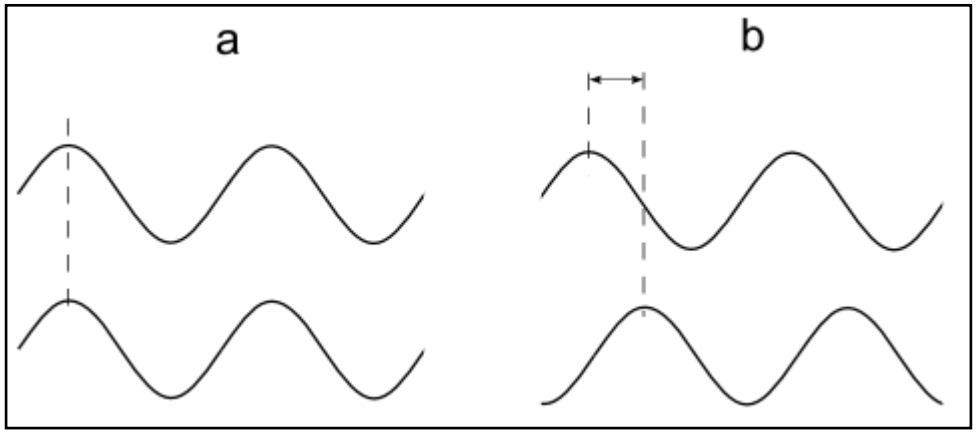

Fig. 4 Differenza di fase
a) nessuna differenza di fase tra le onde
b) onde con differenza di fase

1.2.6. Altre caratteristiche

Oltre alle caratteristiche appena menzionate, possiamo anche fare una distinzione tra la velocità di gruppo e la velocità di fase di un'onda. Le onde possono essere longitudinali o trasversali. Nel caso delle onde trasversali, possiamo anche parlare di polarizzazione. Tali caratteristiche non sono indispensabili per comprendere le questioni legate all'olografia dunque, in questa guida, verranno omesse.

1.3. Proprietà delle onde

Le onde che si propagano nello spazio (ad esempio le onde in acqua, le onde sonore, o le onde elettromagnetiche) presentano alcune proprietà come la riflessione, la rifrazione, la diffrazione e l'interferenza.

1.3.1. Diffrazione

La diffrazione si riferisce a un cambiamento di direzione nella traiettoria di un'onda a causa di un ostacolo o di una fenditura. Questo effetto è ben visibile quando la lunghezza d'onda è comparabile alle dimensioni della fenditura. La Fig. 5 illustra la situazione in cui l'onda incontra la fenditura. Prima che ciò avvenga, l'onda si muove perpendicolarmente alla fenditura (come indicano le frecce rosse). Ci sono anche aree di totale estinzione dell'onda (come indicano le linee blu). Lungo quelle linee l'ampiezza dell'onda è pari a 0.

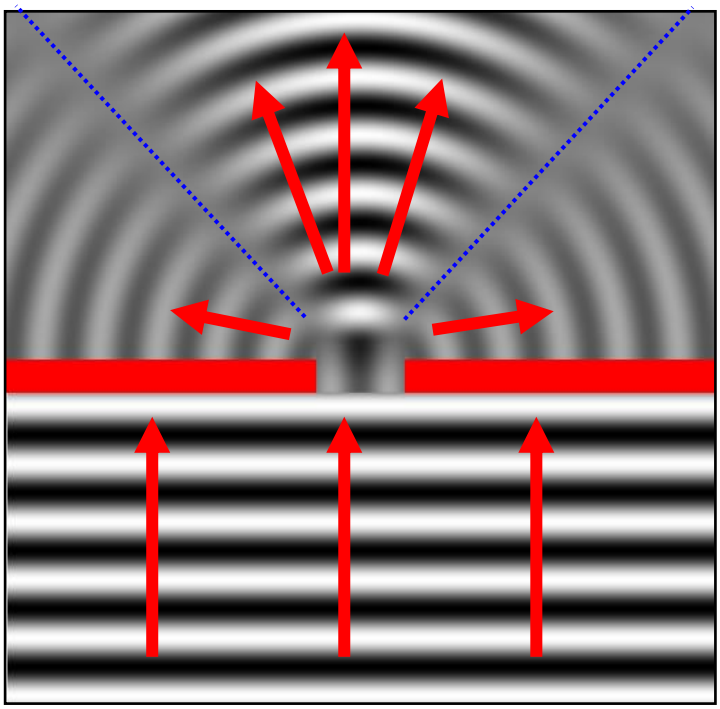

Fig. 5 Diffrazione di una singola fenditura

1.3.2. Interferenza

L'interferenza è il fenomeno su cui si basa l'olografia. Si tratta della sovrapposizione (l'accumularsi) delle onde. Le onde in propagazione interagiscono le une con le altre, rinforzandosi o indebolendosi. La Fig. 6 mostra l'ampliamento e l'estinzione dell'onda. Se le onde sono "in fase" (la differenza di fase è pari a 0) allora si verifica il massimo ampliamento possibile dell'onda. Se, al contrario, le onde sono "fuori fase" per metà del tempo (la differenza di fase è a metà strada nel periodo di propagazione dell'onda), le onde si estinguono del tutto. Quando le onde si rinforzano, l'interferenza è costruttiva. Quando si indeboliscono, è distruttiva.

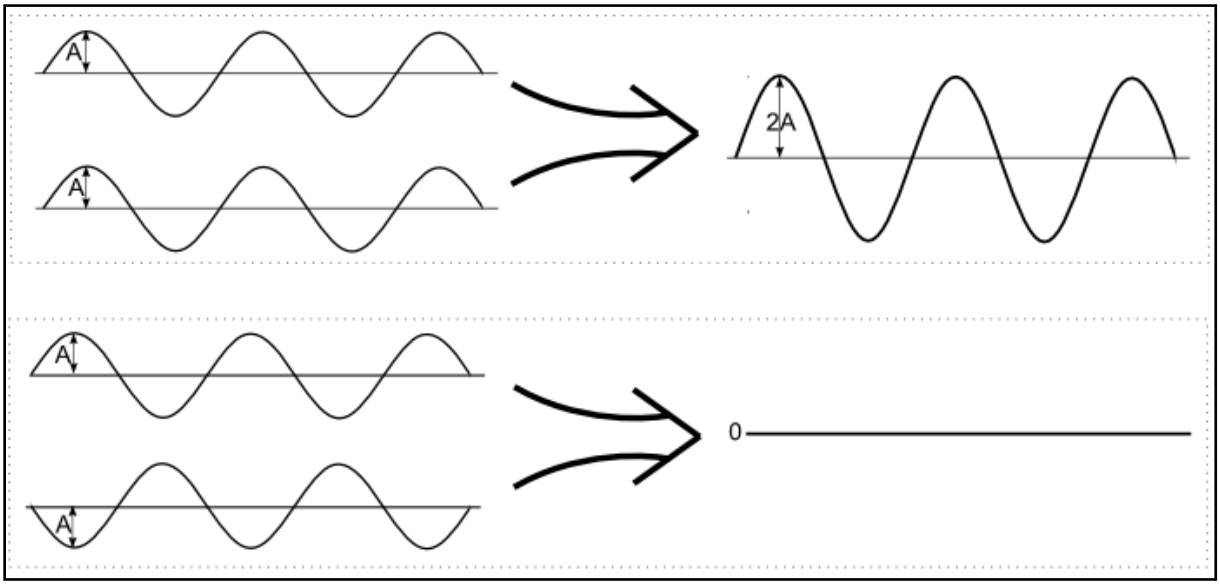

Fig. 6 Interferenza costruttiva e distruttiva

L'interferenza può essere illustrata dall'esempio delle onde sull'acqua. Se simuliamo la superficie dell'acqua in un punto a frequenza costante, otteniamo un'onda che si propaga radialmente. L'effetto viene mostrato nella Fig. 7a. Gli spazi neri rappresentano i ventri dell'onda mentre i gli spazi bianchi, le creste. La Fig. 7b presenta due onde vicine che interagiscono l'una con l'altra. Gli spazi grigi rappresentano l'estinzione dell'onda e quelli non sfuocati coincidono con l'ampliamento dell'onda.

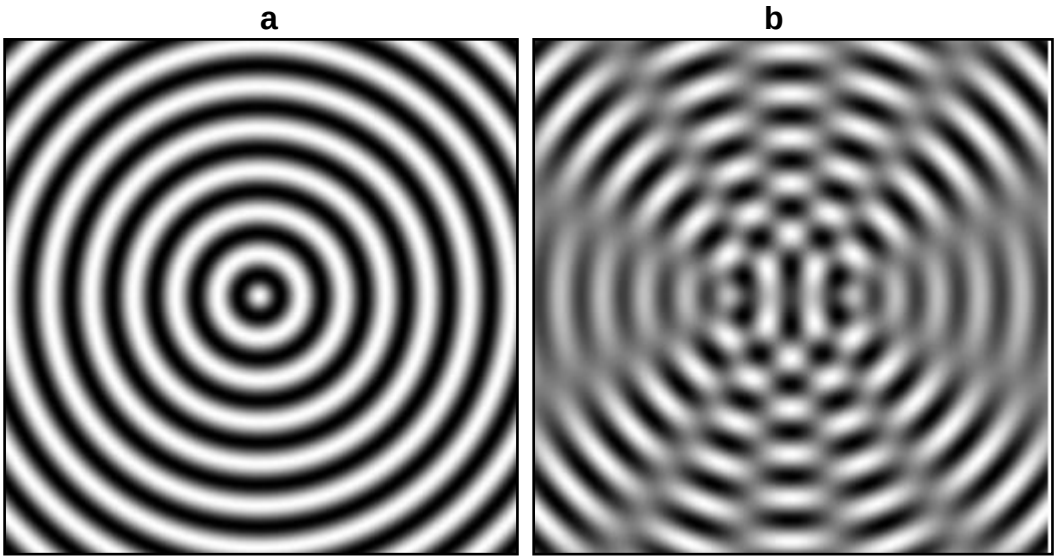

Fig. 7 Un esempio di onde in acqua

1.3.1. Coerenza

La coerenza delle onde è necessaria per ottenere una costante nel tempo del pattern di interferenza. Due onde sono coerenti se hanno una differenza di fase costante. La Fig. 8 mostra, in modo semplificato, come immaginare la differenza tra onde coerenti e incoerenti. Nella parte sopra dell'immagine sono illustrate delle onde ad alta coerenza. Come risultato della sovrapposizione di queste onde, le frange di interferenza formatesi sono costanti nel tempo. Vediamo, dunque, un pattern di interferenza con un buon contrasto (le frange sono chiaramente visibili). Nel caso in cui le onde presentino un basso grado di coerenza, le frange risultano, in ogni momento, allontanate le une dalle altre. I nostri occhi vedono un'immagine sfuocata del pattern di interferenza (con un contrasto ridotto). Più si abbassa il livello di coerenza, più l'immagine sarà sfuocata. Nel caso di onde completamente incoerenti, non vedremo nessun pattern di interferenza. La maggior parte della luce in natura è formata da onde incoerenti. Ad ogni modo, una buona fonte di onde coerenti è il laser.

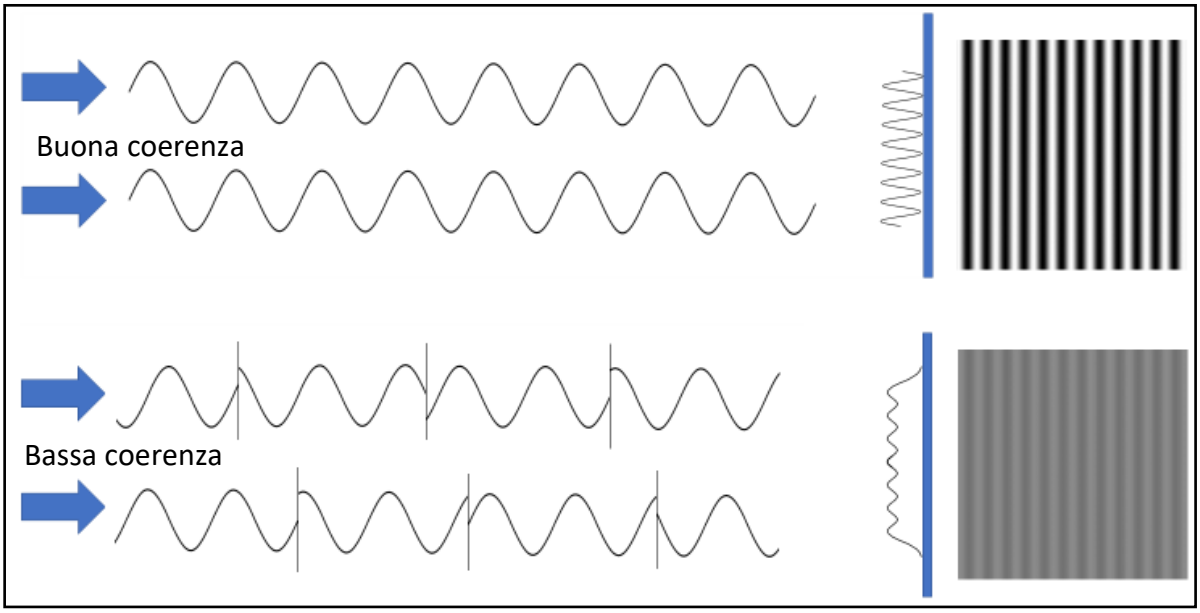

Fig. 8 Una buona coerenza è necessaria a ottenere un alto contrasto nel pattern di interferenza

1.4. Onde in 3D

1.4.1. Onda sferica e onda piana

L'onda in 3D può essere rappresentata come una superficie in cui la fase è costante. L'esempio più comune è quello di un'onda sferica o piana (Fig. 9). Nel primo caso (Fig. 9a), le superfici della fase costante (vale a dire le superfici in cui l'onda ha lo stesso valore, ad esempio raggiunge il valore massimo) prendono la forma di sfere (e di qui il nome di onda sferica). Nel secondo caso (Fig. 9b), le superfici della fase costante prendono la forma di piani (di qui il nome di onda piana).

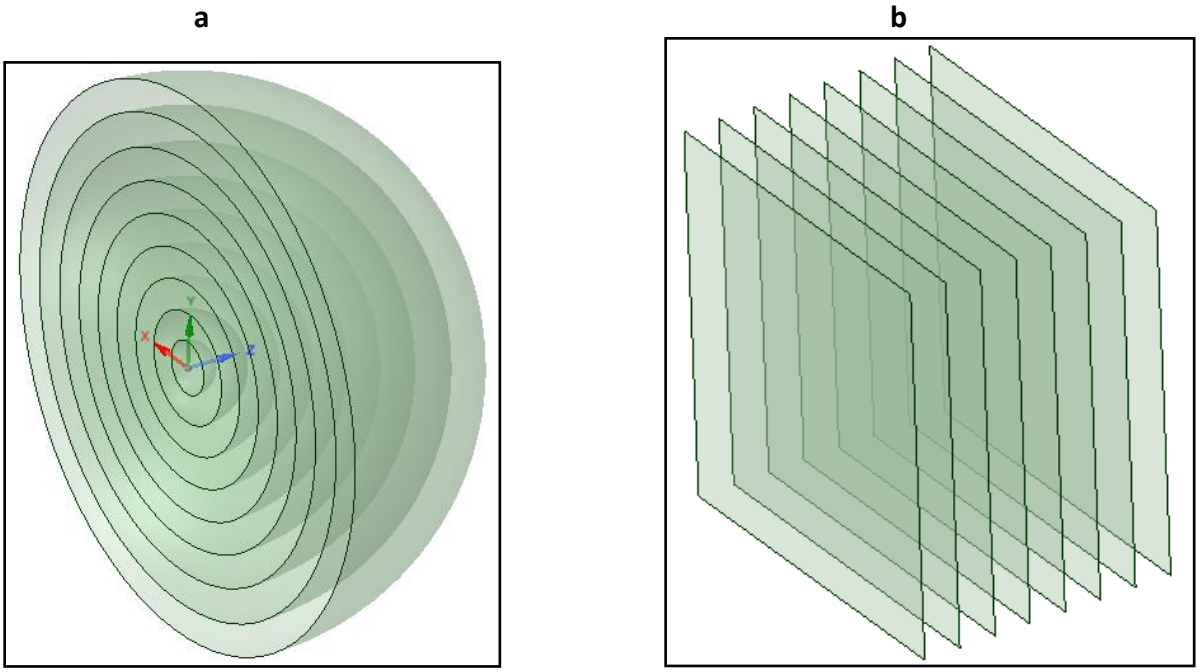

Fig. 9 Onda sferica (a) e onda piana (b)

1.4.2. Principio di Huygens–Fresnel

Consideriamo il caso in cui le onde toccano un oggetto (Fig. 10). Possiamo assumere che ogni punto toccato dall'onda diventa la sorgente di una nuova onda sferica. La somma di tutte queste onde sferiche (ossia l'interferenza) determina la forma della nuova onda. Nella Fig. 10 l'onda incidente è segnata in nero. L'oggetto raggiunto dall'onda può essere rappresentato come una serie di infiniti punti che, nella figura, coincidono con i pallini verdi. Ognuno di questi punti è la sorgente di un'onda sferica secondaria. Sommando tutte le onde sferiche secondarie, otteniamo una nuova onda (tratteggiata in rosso nella figura) che poi si propaga nello spazio. Si tratta del principio di Huygens-Fresnel, che recita così:

Ogni punto raggiunto da un'onda diventa la sorgente di un'onda sferica; la somma di queste onde secondarie determina la forma dell'onda in qualunque momento successivo.

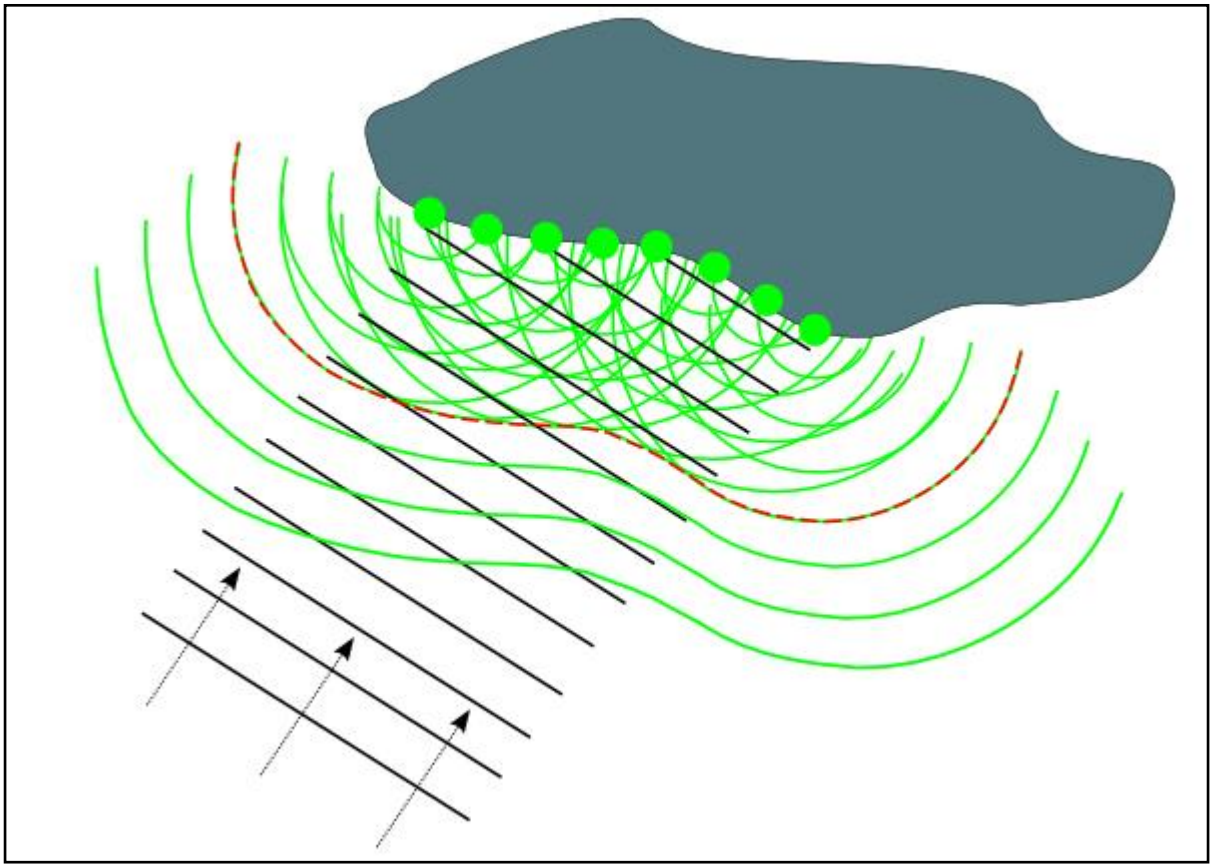

Fig. 10

1.5. La luce si comporta come un'onda

Anche la luce è un'onda. Se è così, allora che cos'è che ondeggia? Ciò che ondeggia è il campo magnetico. Per questa ragione si dice che la luce è un'onda elettromagnetica (EM). Tale onda può essere visualizzata come l'aumentare e il diminuire dei campi elettrici e magnetici che si propagano nello spazio (Fig. 11). Il campo elettrico è evidenziato in rosso e quello magnetico in blu.

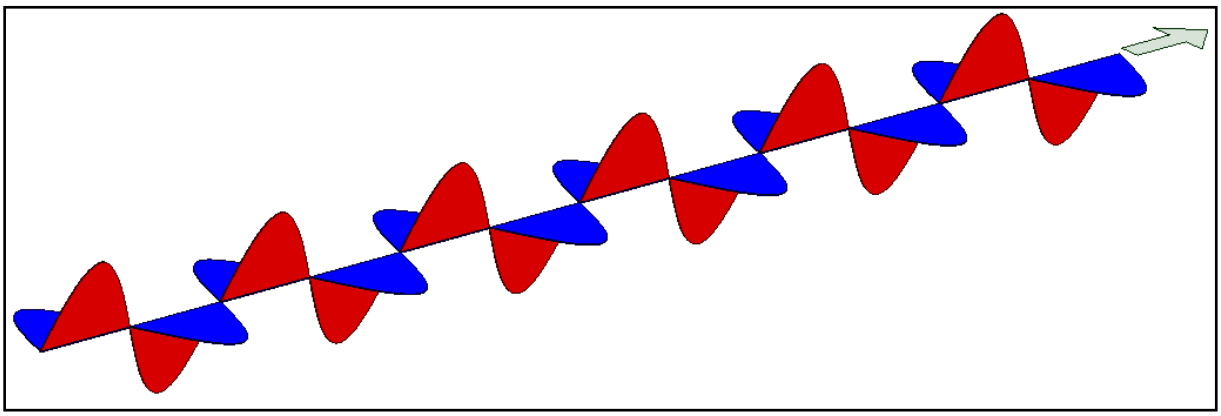

Fig. 11 Onda elettromagnetica

In natura, esistono molti tipi di radiazione elettromagnetica. Sono classificati in base alla velocità in cui i campi elettromagnetici cambiano nello spazio (ossia in base alla frequenza delle onde). Le onde EM con la frequenza più alta sono raggi gamma, poi abbiamo i raggi X, i raggi ultravioletti, lo spettro visibile, i raggi infrarossi, le microonde e le onde radio, radiazioni EM con la frequenza più bassa (Fig. 12).

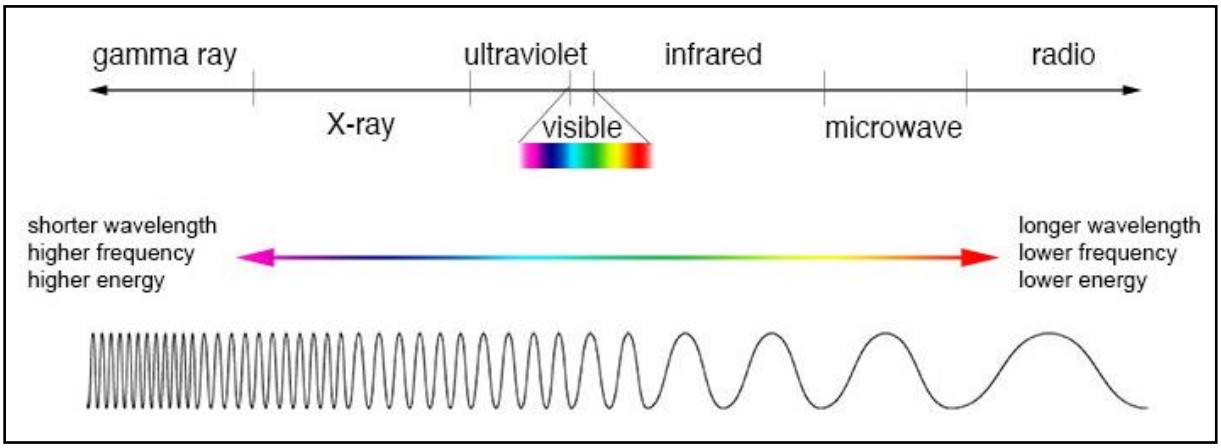

Fig. 12 Lo spettro delle onde elettromagnetiche. Fonte: NASA, Imagine the Universe (https://imagine.gsfc.nasa.gov/science/toolbox/emspectrum1.html)

2. ___

2.1. L'olografia in parole povere

L'ologramma è una lastra o una lastra di vetro sulla cui superficie o volume viene registrato un pattern d'onda interferente. La luce che passa attraverso o che si riflette dall'ologramma crea un'immagine identica alla luce riflessa dall'oggetto reale (vedi Fig. 13).

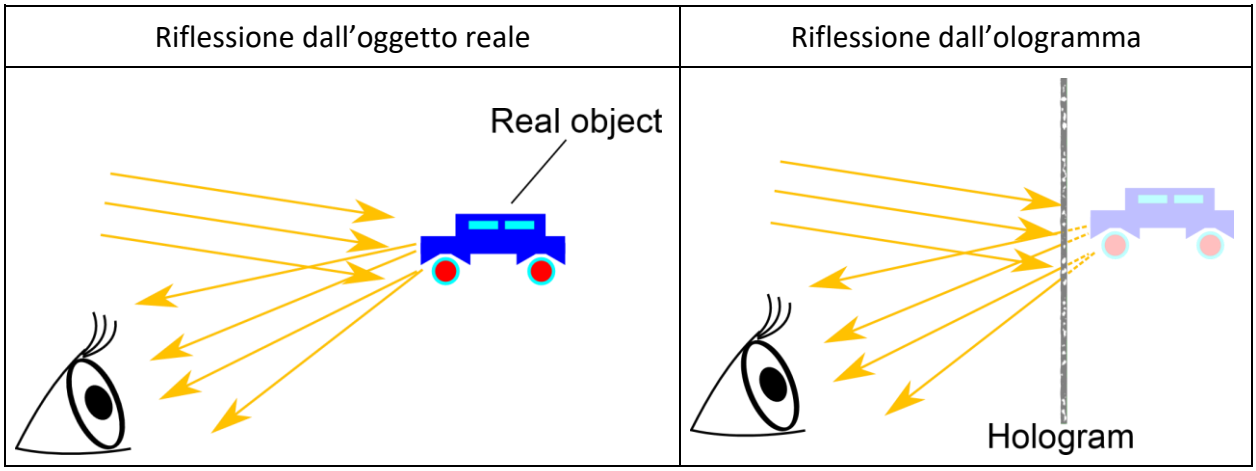

Fig. 13 La luce riflessa dall'ologramma si comporta esattamente come se provenisse dall'oggetto reale

Un elemento caratteristico dell'ologramma è che l'oggetto osservato è completamente tridimensionale. Questo significa che guardando un ologramma da una certa angolazione, possiamo scorgere dei dettagli che sarebbero invisibili se lo guardassimo da un'altra (see Fig. 14). Questa particolare proprietà non riguarda invece le normali fotografie; da qualunque angolo d'osservazione le si guardi, le fotografie appariranno sempre uguali.

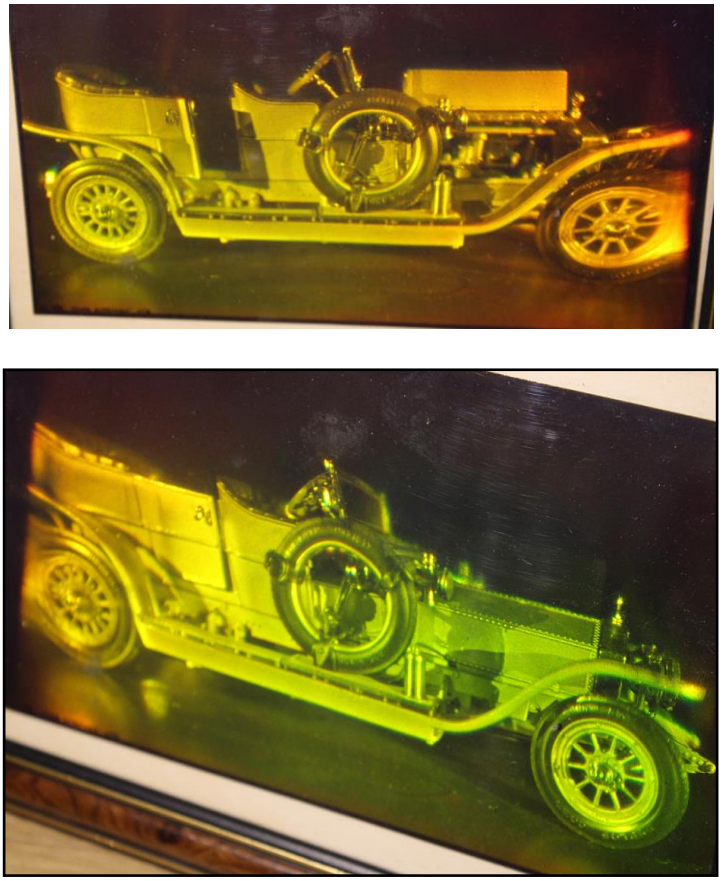

Fig. 14 L'immagine dell'ologramma cambia in base all'angolo d'osservazione.

2.2. Principio olografico

Il fenomeno fisico che crea l'ologramma è l'interferenza. Sappiamo già che due onde coerenti interferiscono l'una con l'altra creando frange d'interferenza. Un esempio semplice è quello della Fig. 15, che mostra due onde piane che interferiscono tra loro creando delle frange dritte.

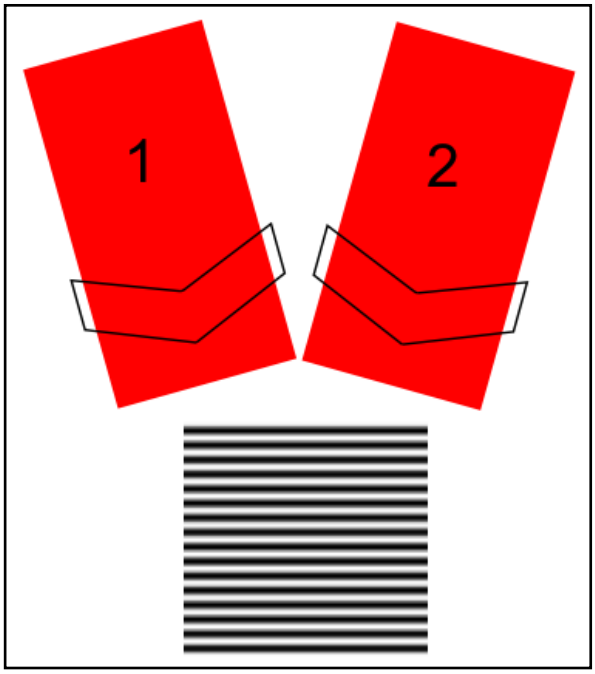

Fig. 15 Interferenza di due onde piane

L'interferenza di un'onda piana con una sorgente puntiforme è illustrata nella Fig. 16. Le frange d'interferenza hanno la forma di cerchi. Possiamo dire che tale pattern di interferenza contenga informazioni su di un punto nello spazio. Se riusciamo a registrarne la distribuzione, ad esempio su una pellicola olografica, allora è possibile ricostruire il punto decodificato illuminando il pattern con un'onda piana.

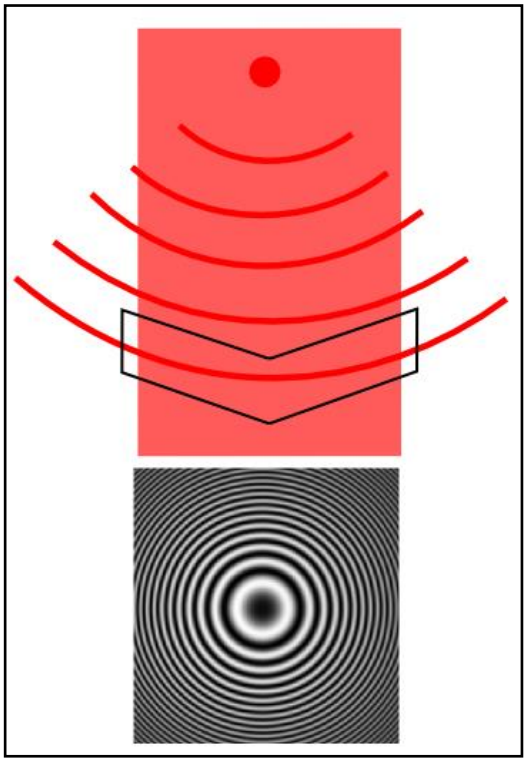

Fig. 16 L'interferenza di un'onda piana con una sorgente puntiforme

Ogni oggetto illuminato può essere trattato come un'ampia serie di sorgenti puntiformi (principio di Huygens-Fresnel). Il pattern di interferenza risultante dall'interferenza di molte sorgenti puntiformi con un'onda piana di solito è complesso (vedi Fig. 17). Possiamo dire che l'oggetto si codifica come pattern di interferenza.

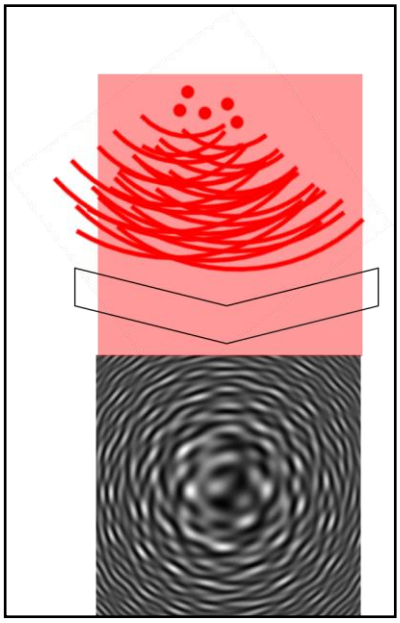

Fig. 17 L'interferenza di un'onda piana con molte sorgenti puntiformi

Quel pattern è un ologramma. L'ologramma può essere ricostruito illuminandolo con un'onda piana. La Fig. 18 mostra un esempio di registrazione e ricostruzione di un ologramma. In questo caso, durante la ricostruzione dell'ologramma, otteniamo una vera immagine (visibile sullo schermo) e un'immagine virtuale (visibile quando guardiamo direttamente l'ologramma illuminato).

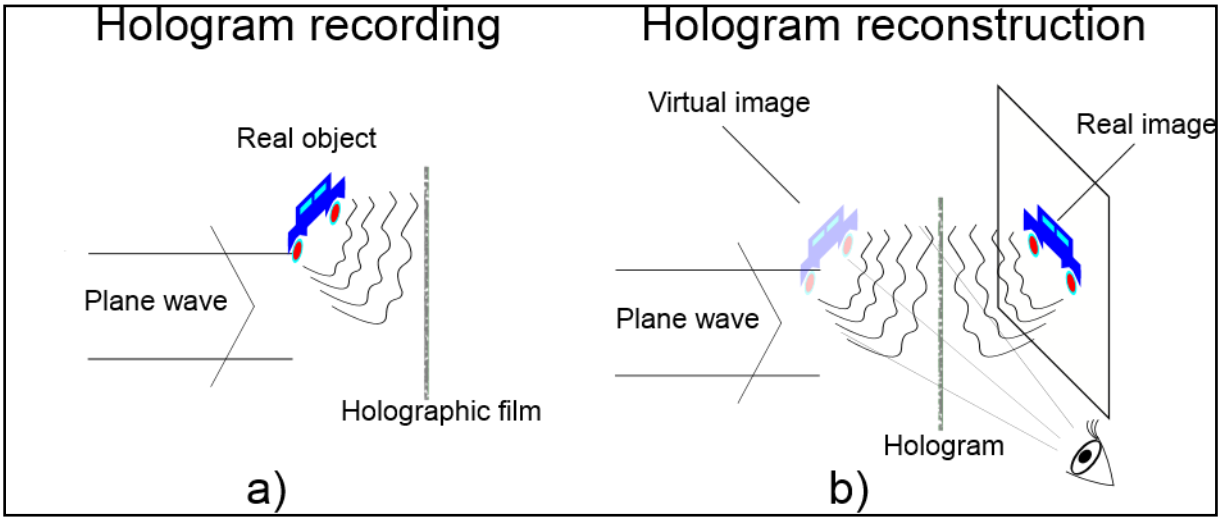

Fig. 18 Registrazione (a) e ricostruzione di un ologramma (b)

2.3.Tipi di ologramma e proprietà

2.3.1. Tipi di ologramma

Gli ologrammi possono essere classificati in vari modi in base alle loro proprietà. Una di queste è la modalità in cui la luce li attraversa. Sappiamo che l'ologramma può essere modulato ad ampiezza o a fase. La modulazione ad ampiezza si verifica quando il pattern è registrato nella forma di aree più o meno trasparenti(Fig. 19a). L'ologramma modulato a fase è invece completamente trasparente e il pattern è registrato nella forma di aree con diverso indice di rifrazione (Fig. 19b).

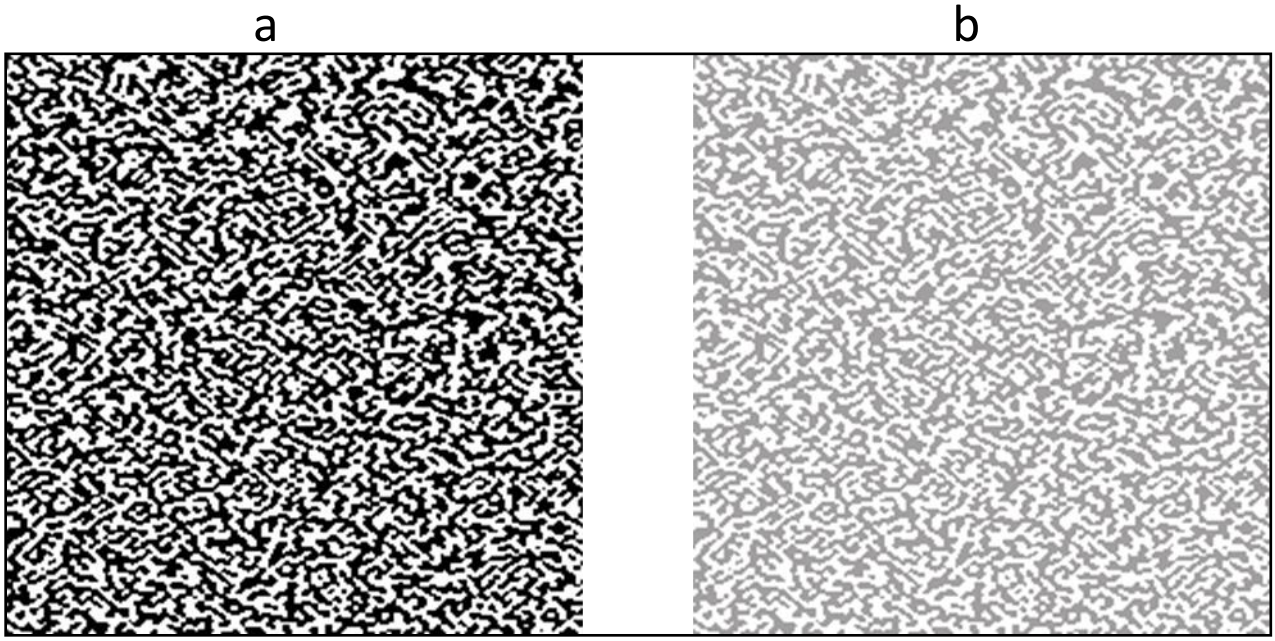

Fig. 19 Ologramma modulato ad ampiezza (a) e fase (b)

Gli ologrammi possono anche essere suddivisi tra sottili e spessi. Gli ologrammi sottili sono quelli che si comportano come se il pattern fosse registrato solo sulla superficie della lastra olografica. Gli ologrammi spessi sono quelli in cui il pattern è registrato sull'intero volume dell'ologramma.

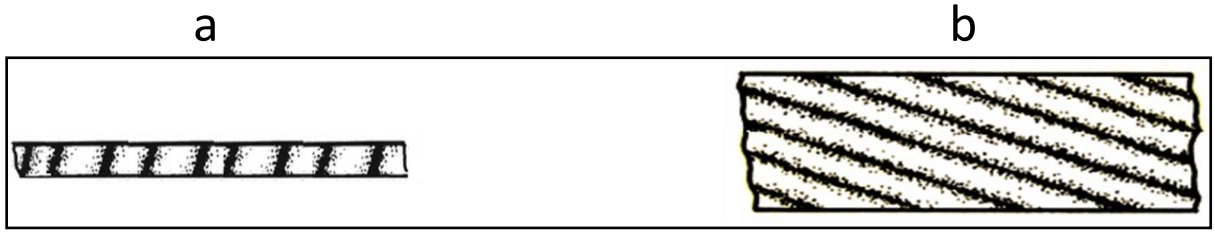

Fig. 20 Ologramma sottile (a) e spesso (b)

Gli ologrammi si distinguono anche tra ologrammi a trasmissione e a riflessione. L'ologramma a trasmissione può essere osservato trasmettendo la luce attraverso l'ologramma stesso, mentre quello a riflessione si osserva riflettendo la luce proveniente dall'ologramma.

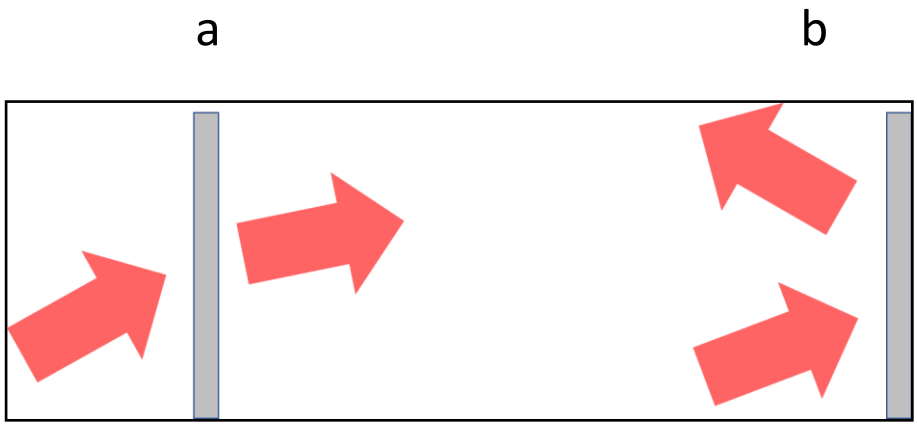

Fig. 21 Ologramma a trasmissione (a) e a riflessione (b)

2.3.1. Proprietà degli ologrammi

L'olografia è una tecnologia fotografica – le informazioni riguardo all'immagine vengono registrate sulla lastra olografica. Tuttavia, esistono differenze fondamentali tra ologrammi e comuni fotografie:

- Nel caso dell'olografia, la pellicola deve avere una risoluzione molto più alta (misurata in linee per millimetro).
- Il negativo fotografico contiene soltanto informazioni a proposito dell'intensità della luce, mentre l'ologramma (in forma di pattern di interferenza) contiene informazioni sia sull'intensità della luce che sulla fase (cioè sulla distanza dei punti dell'oggetto dall'ologramma).
- Ogni frammento del negativo fotografico contiene informazioni su una specifica porzione di immagine mentre, nel caso dell'ologramma, ogni frammento contiene informazioni sull'intera immagine. Ciò significa che se tagliamo l'ologramma in due parti, ognuna potrà ricostruire l'immagine completa. Tuttavia, ci sarà una differenza di luminosità e il campo di osservazione da cui guardiamo l'ologramma sarà diminuito.

2.4. Strumenti basici per la registrazione degli ologrammi

2.4.1. Ologramma di Gabor

Storicamente, il primo strumento per la registrazione degli ologrammi fu proposto da Denis Gabor nel 1948. I laser, all'epoca, non erano ancora stati inventati. Questo strumento usa una sorgente di luce grazie al quale risulta parzialmente coerente e registrabile come ologramma. L'oggetto è una pellicola trasparente (con un segno o un disegno semplice). Un po' di luce passa direttamente attraverso la pellicola e un po' si disperde. Entrambi questi

raggi interferiscono l'uno con l'altro registrandosi sulla pellicola come pattern di interferenza (vedi Fig. 22).

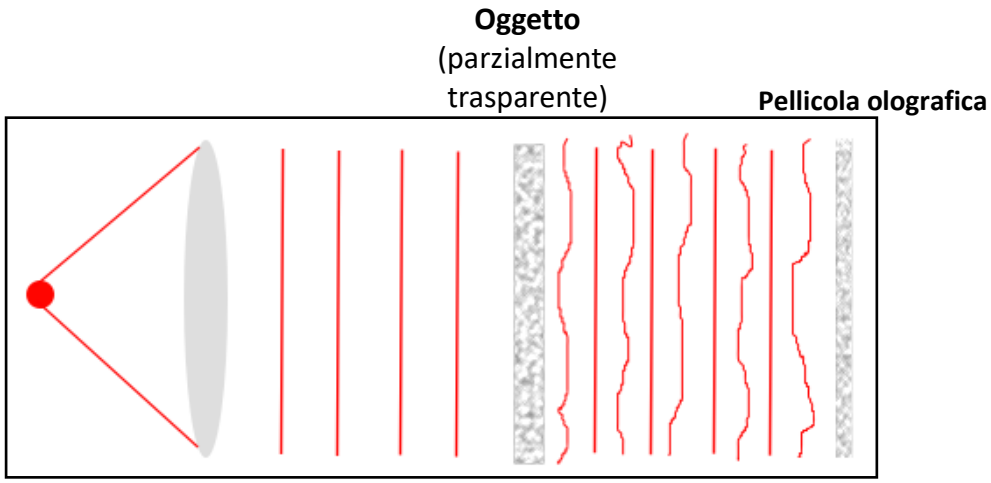

Fig. 22 Ologramma di Gabor

2.4.1. Lo strumento di Leith-Upatnieks

Con questo strumento è possibile registrare il cosiddetto ologramma di Fresnel che può essere ricostruito con la luce del laser. Il raggio laser è diviso in due (raggio riferimento e raggio oggetto). La luce si disperde quando l'oggetto interferisce con il raggio riferimento e lo registra come pattern di interferenza sulla pellicola olografica (Fig. 23). La Fig. 24 illustra un esempio dell'ologramma di Fresnel.

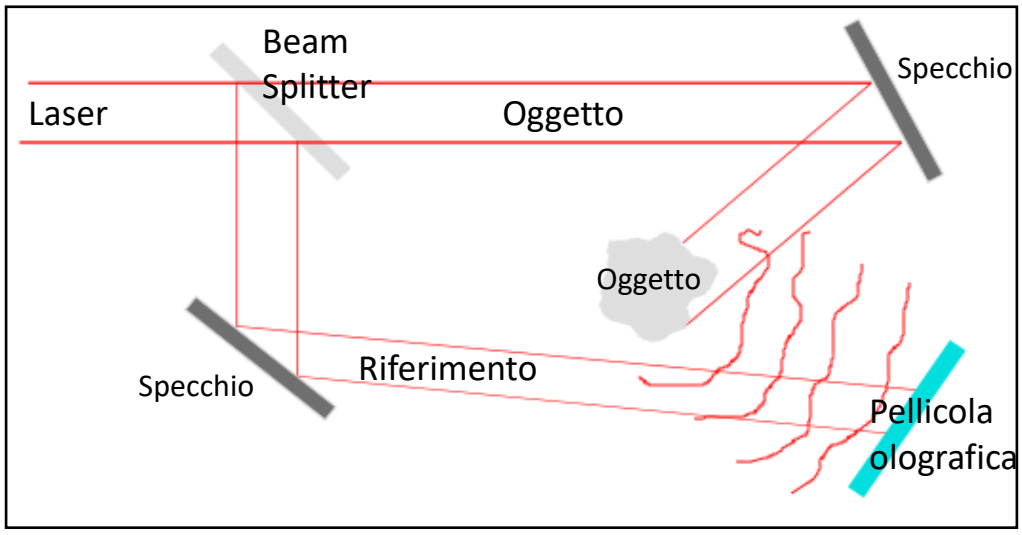

Fig. 23 Ologramma di Fresnel registrato con lo strumento di Leith-Upatnieks

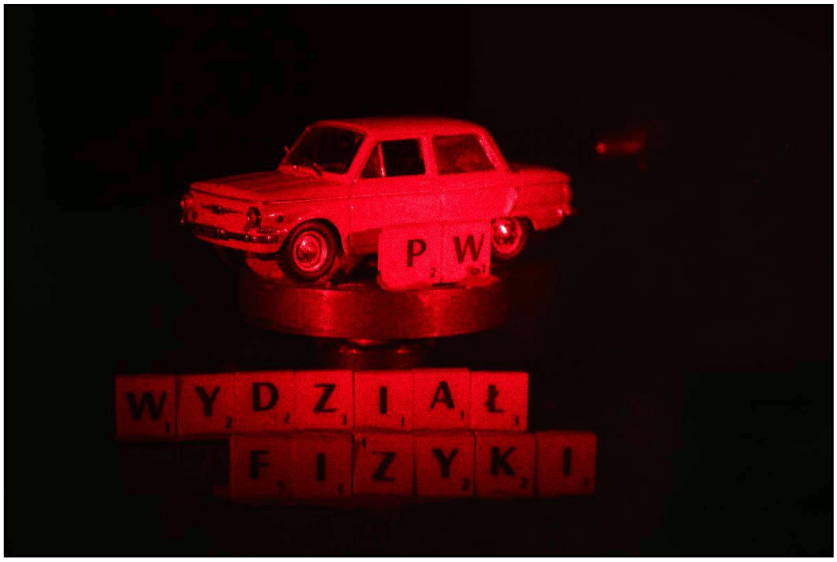

Fig. 24 Un esempio di ologramma di Fresnel

2.4.1. Ologramma arcobaleno (Benton)

Con questo metodo (Fig. 25), un ologramma di Fresnel (precedentemente illustrato nel sistema di Leith-Upatnieks) è usato come oggetto. L'ologramma ottenuto ha il vantaggio di poter essere visto alla luce bianca. Girando delicatamente l'ologramma in una direzione, l'effetto 3D diventa visibile (la cosiddetta parallasse) mentre nell'altra direzione si può osservare il cambiamento dei colori. L'ologramma arcobaleno è esemplificato nella Fig. 26.

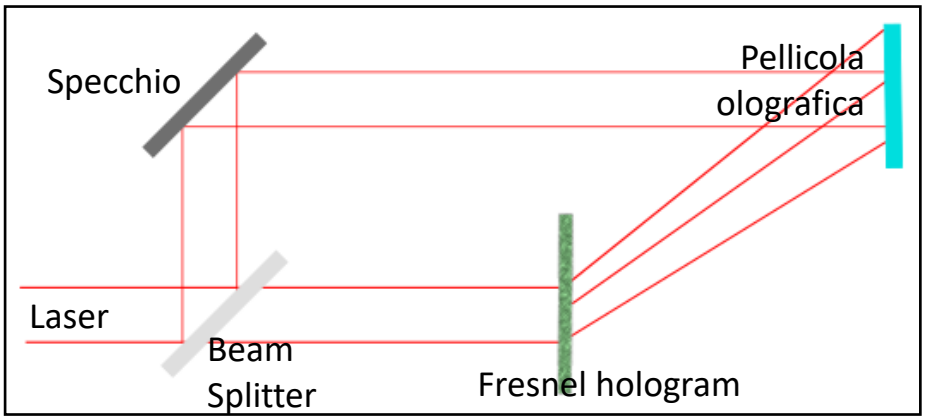

Fig. 25 Registrare un ologramma arcobaleno

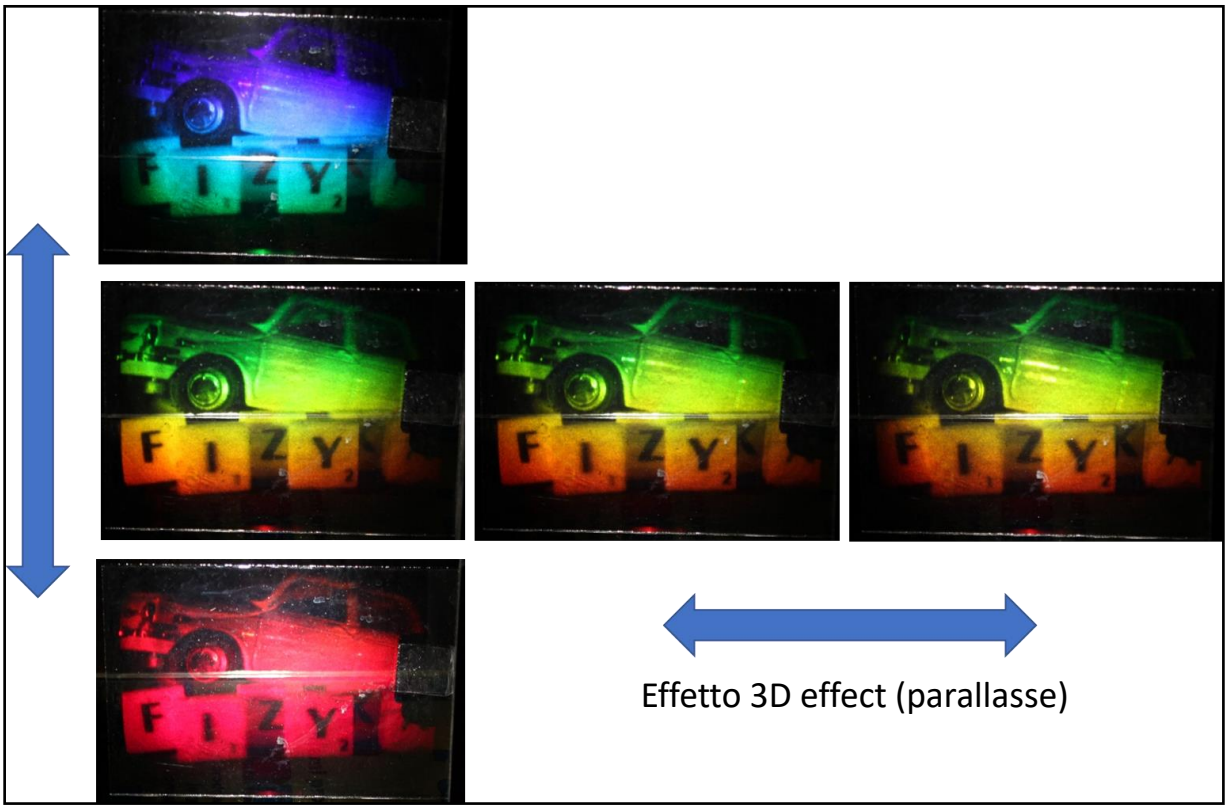

Fig. 26 Un esempio di ologramma arcobaleno

2.4.1. Ologramma di volume

Questo ologramma può essere registrato con il metodo illustrato dalla Fig. 27. L'ologramma di volume è visibile a trasmissione sotto luce bianca. Un esempio è quello della Fig. 28.

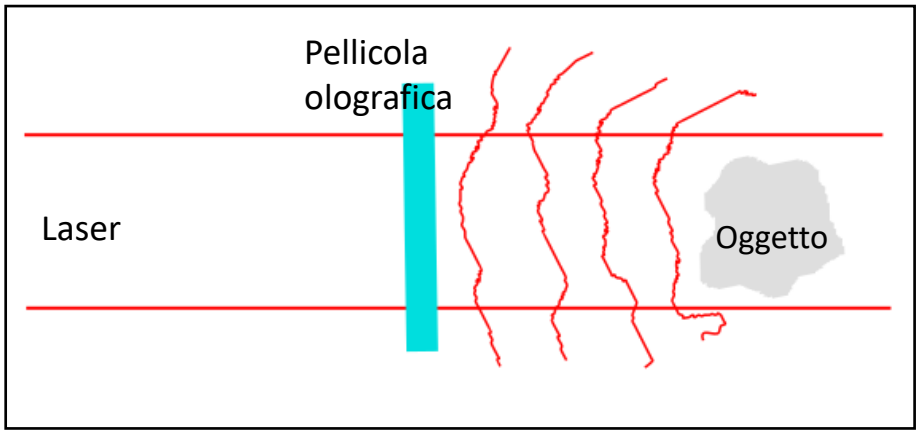

Fig. 27 Metodo per la registrazione dell'ologramma di volume

Fig. 28 Un esempio di ologramma di volume

2.4.1. Ologrammi generati a computer

Non c'è bisogno di nessun oggetto fisico per generare un ologramma dal computer (CGH). Nella versione più semplice, l'oggetto consiste in un file grafico con un segno o un'immagine a due dimensioni. Sulla base di questo file, un pattern di interferenza (anche in questo caso un file grafico) viene computato dal computer. Per i calcoli, viene utilizzato un algoritmo interattivo (quello di Gerchberg-Saxton è il più popolare). Il pattern di interferenza calcolato può essere realizzato nella forma di elemento ottico (slide). Se illuminiamo le slide con un raggio laser (per esempio con un puntatore laser) allora la luce creerà una forma precostituita. Un esempio di tale ologramma potrebbero essere i popolari filtrini per il laser che generano una forma diversa rispetto al puntino. Realizzare un ologramma ad alta qualità richiede un equipaggiamento sofisticato. Tuttavia, anche in un contesto domestico può essere creato un CGH per scopi educativi. A tal fine, il pattern di interferenza calcolato dovrebbe essere stampato, per esempio, su un foglio formato A4. Poi bisognerebbe fotografarlo con una fotocamera analogica. Dopodiché tutto quello che bisogna fare è sviluppare la pellicola, per ottenere un ologramma generato a computer. Il procedimento per realizzare questi ologrammi è esemplificato nella Fig. 29.

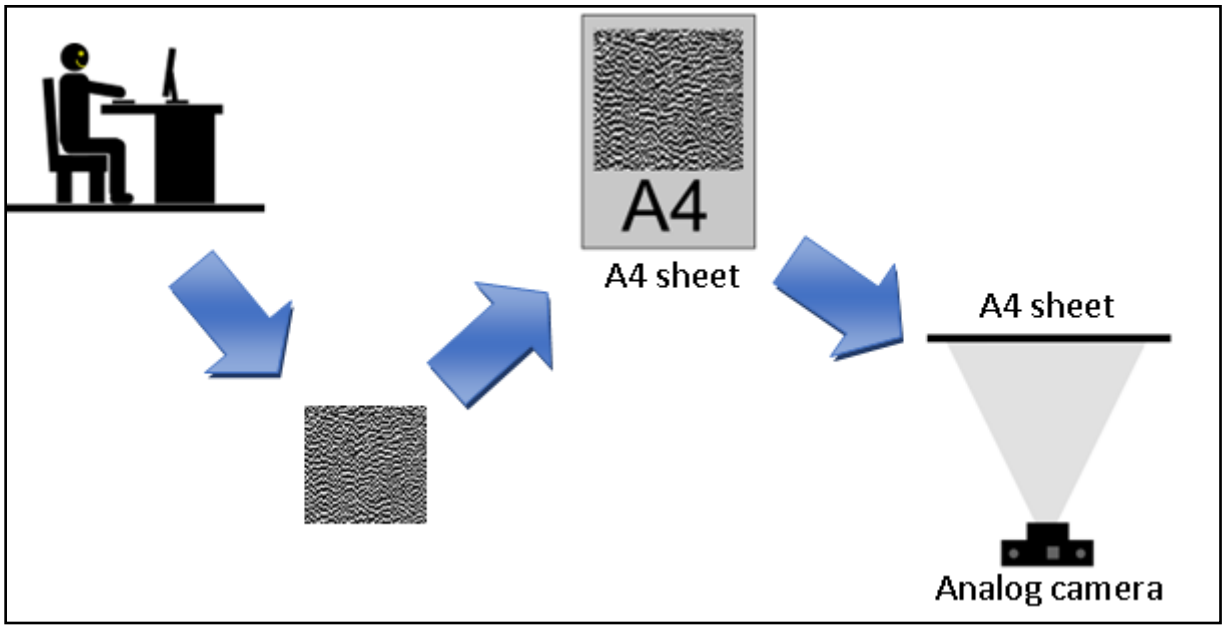

Fig. 29 Procedimento per ottenere un CGH a casa

3. Gestione delle immagini con il programma Octave

3.1. Istallazione di Octave

Il programma Octave dev'essere scaricato dal sito:
https://www.gnu.org/software/octave/
Per farlo, cliccate sul pulsante di download come nella Fig. 30:

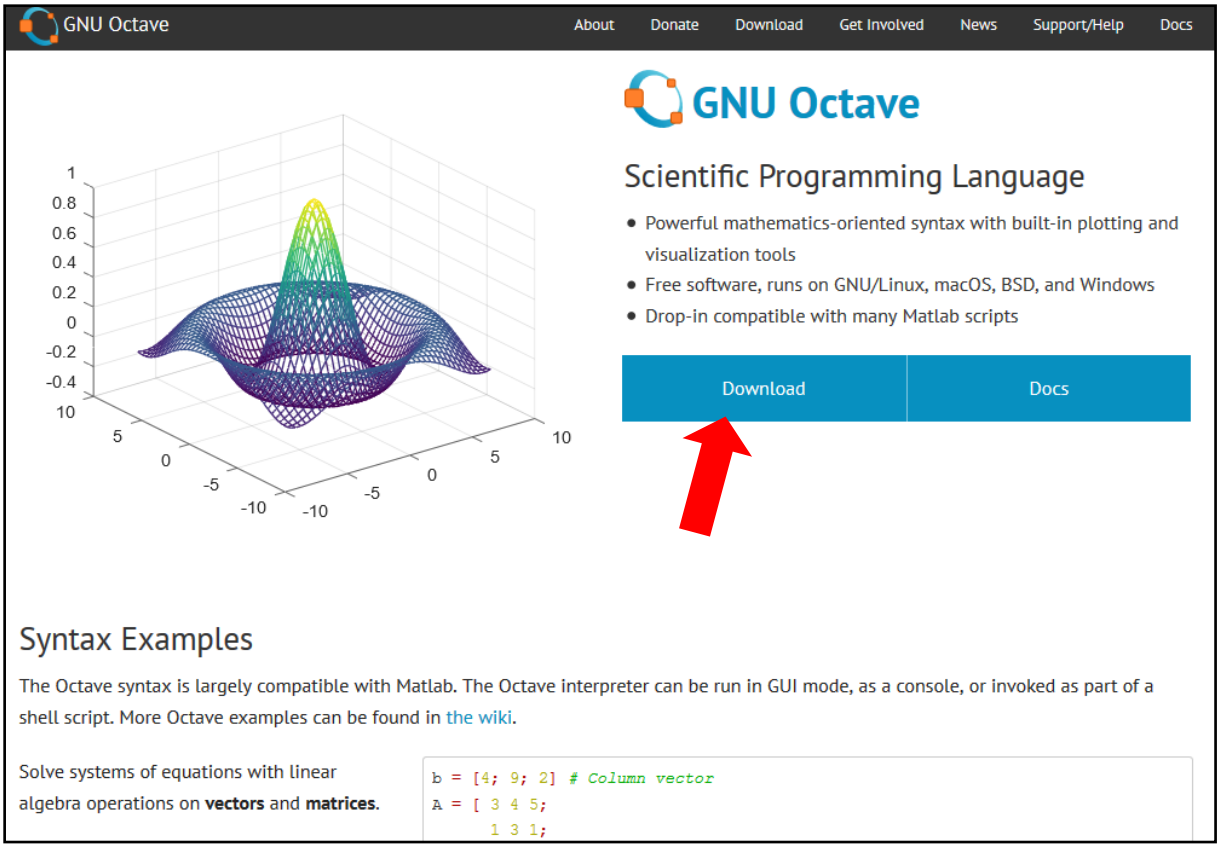

Fig. 30 Scaricare GNU Octave

Poi andate sulla barra di Windows e scaricate il file: octave-4.2.1-w32-installer.exe se dovete scaricare la versione da 32 bit, oppure: octave-4.2.1-w64-installer.exe per la 64 bit (Fig. 31).

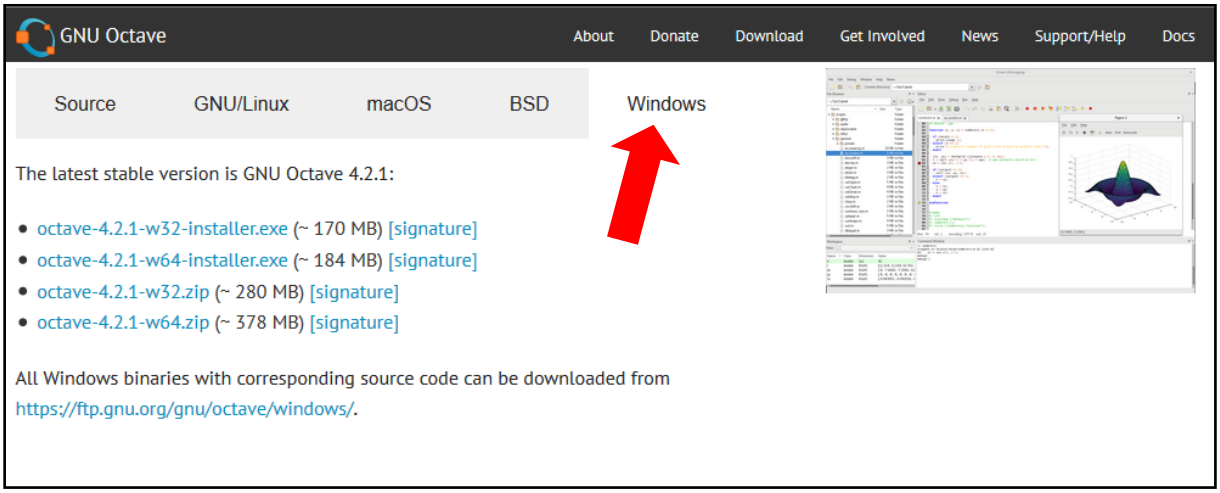

Fig. 31 Scaricare GNU Octave

3.2. Azionare il programma

Se lavorate con Windows, cliccate sul menù Start e scrivete "Octave" nella ricerca. Poi aprite il file "Octave (GUI)" . Apparirà la schermata principale, formata da diverse finestre (Fig. 32), la più importante delle quali è l'area dove inserire i comandi (la numero 1 nella figura), poi una finestra che permette di scegliere una cartella in cui lavorare (numero 2) e infine un browser di file nella presente cartella (numero 3).

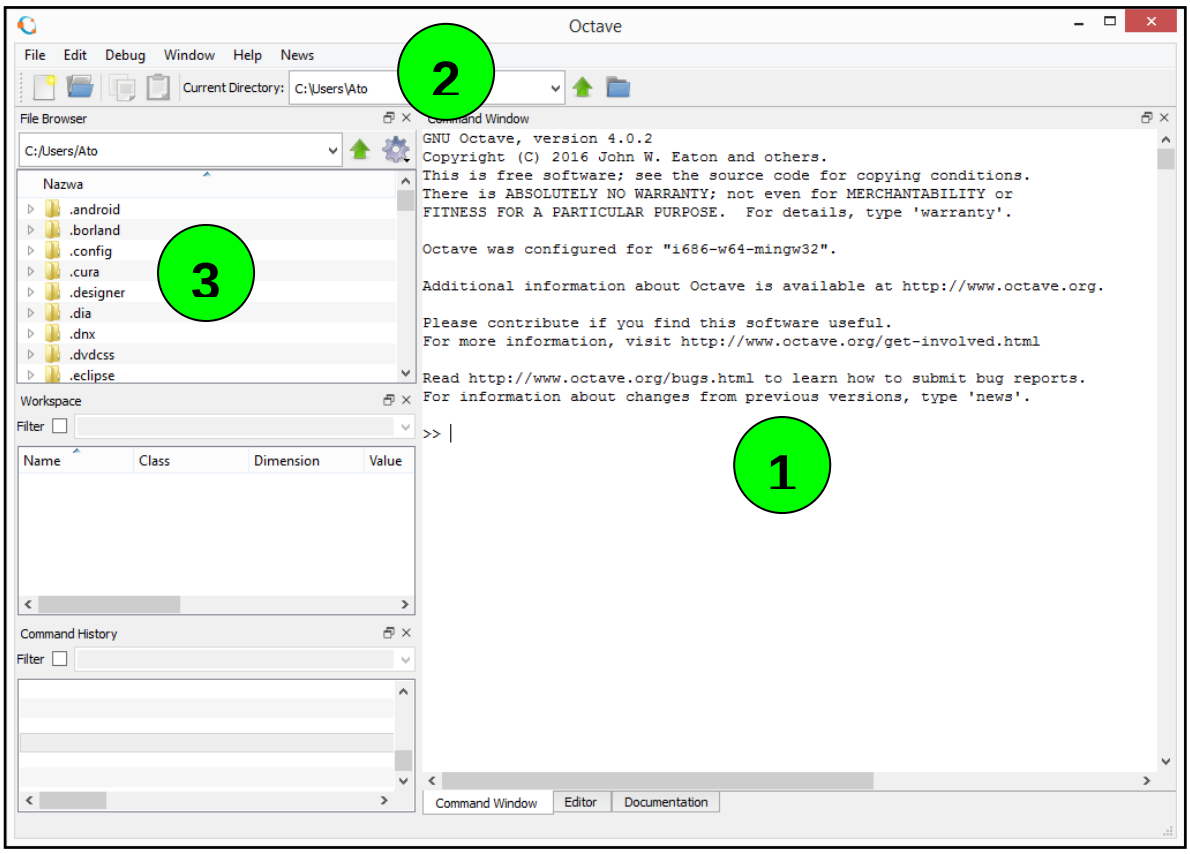

Fig. 32 Azionare Octave

3.3. Finestra dei comandi

Ci sono due modi per inserire i comandi: con la funzione "Command Window" e con la funzione "Editor". Nel primo caso, ogni comando viene eseguito immediatamente dopo aver premuto il tasto Invio. Nel secondo, il cosiddetto script deve essere digitato (si tratta di una serie di comandi), dopodiché i comandi di ogni riga verranno eseguiti uno dopo l'altro. Adesso potrà sembrare incomprensibile, ma in seguito troverete alcuni esempi.

Nella Fig. 33 viene mostrato come passare dalla command window all'editor.

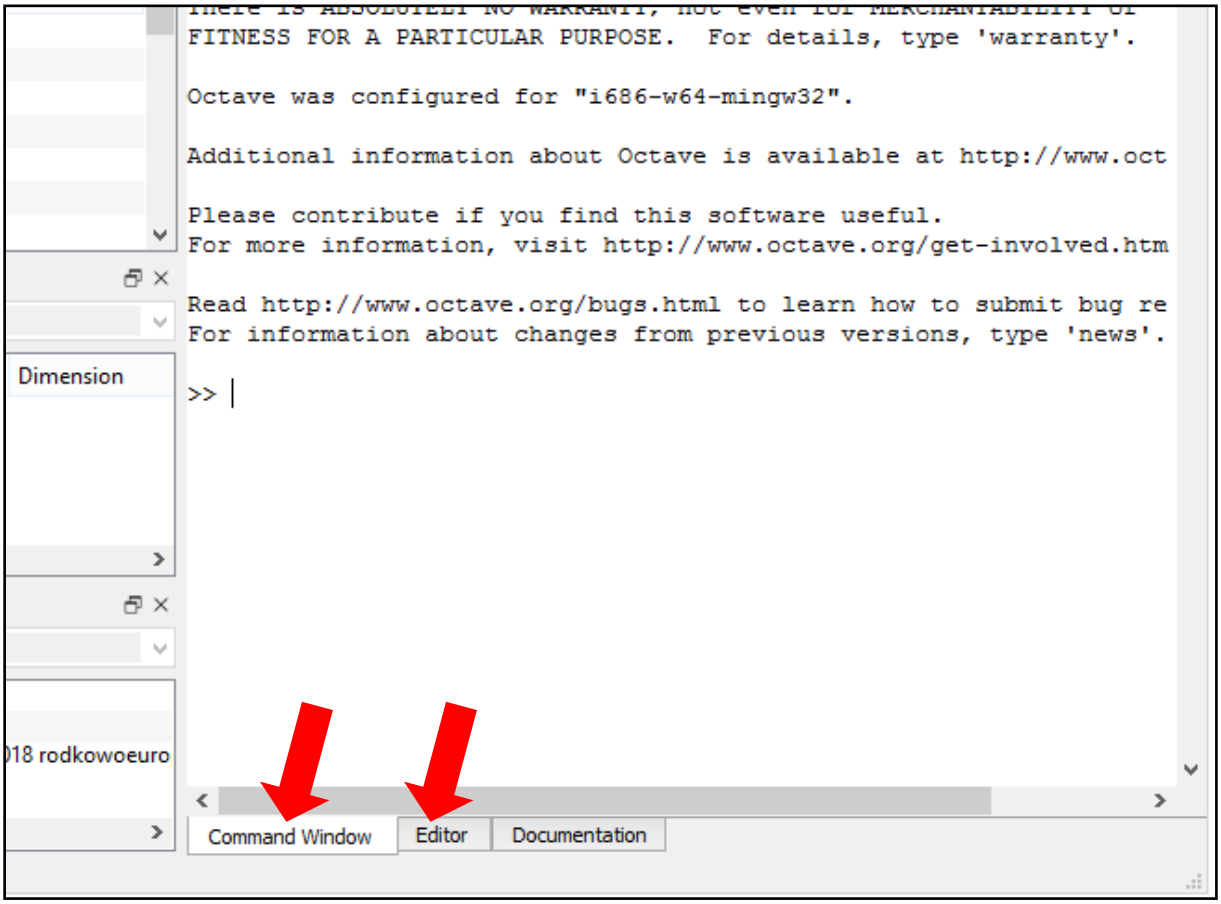

Fig. 33 Command Window e Editor

La Command Window si usa per i calcoli brevi e veloci. Ad esempio, se vogliamo stabilire il valore dell'espressione $\cos^2\left(\frac{\pi}{4}\right)$ digitiamo nella finestra l'espressione `(cos(pi/4))^2`. L'operatore ^ serve a elevare a potenza (nel nostro caso alla seconda). Dopo aver inserito l'espressione qui sopra e aver confermato con il tasto Invio, otterremo immediatamente il risultato, come mostrato dalla Fig. 34.

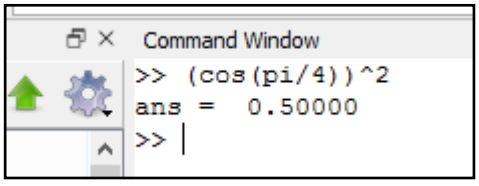

Fig. 34 Risultato del calcolo

Il risultato viene dato automaticamente con il nome della variabile "ans". Possiamo utilizzare questa variabile in modo molto semplice per i calcoli futuri. Ad esempio, se vogliamo che il nostro risultato sia ulteriormente elevato alla seconda, semplicemente digitiamo sulla tastiera l'espressione `ans ^ 2` come mostrato dalla Fig. 35. Poi la variabile "ans" assumerà un nuovo valore, in questo caso uguale a 0.25.

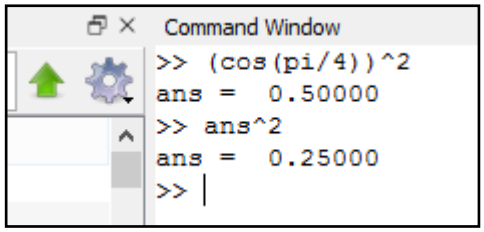

Fig. 35 Calcoli successivi

Ovviamente possiamo dare valori diversi a variabili create indipendentemente. Se, ad esempio, volessimo dare un valore di 5 a una variabile che chiamiamo "a", allora dobbiamo scrivere solo a = 5 come mostrato dalla Fig. 36.

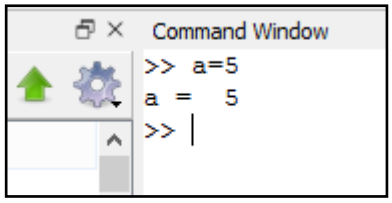

Fig. 36 Impostazione della variabile

In questo modo impostiamo una nuova variabile chiamata "a" e allo stesso tempo le assegniamo il valore di 5. Dopo aver premuto il tasto Invio, il risultato del nostro comando appare sullo schermo (`a = 5`). Se non vogliamo che il risultato appaia, il comando deve essere terminato con un punto e virgola, come mostrato qui:

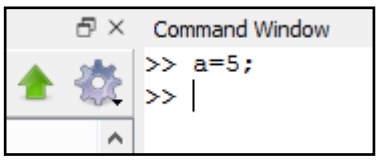

Fig. 37 Il risultato del comando non compare esplicitamente.

In questa maniera potete creare diverse variabili ed eseguire molteplici calcoli (Fig. 38).

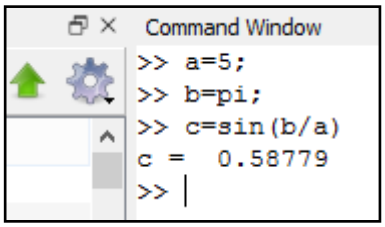

Fig. 38 I calcoli

Tutte le variabili impostate sono visibili nella finestra “Workspace" (Fig. 39).

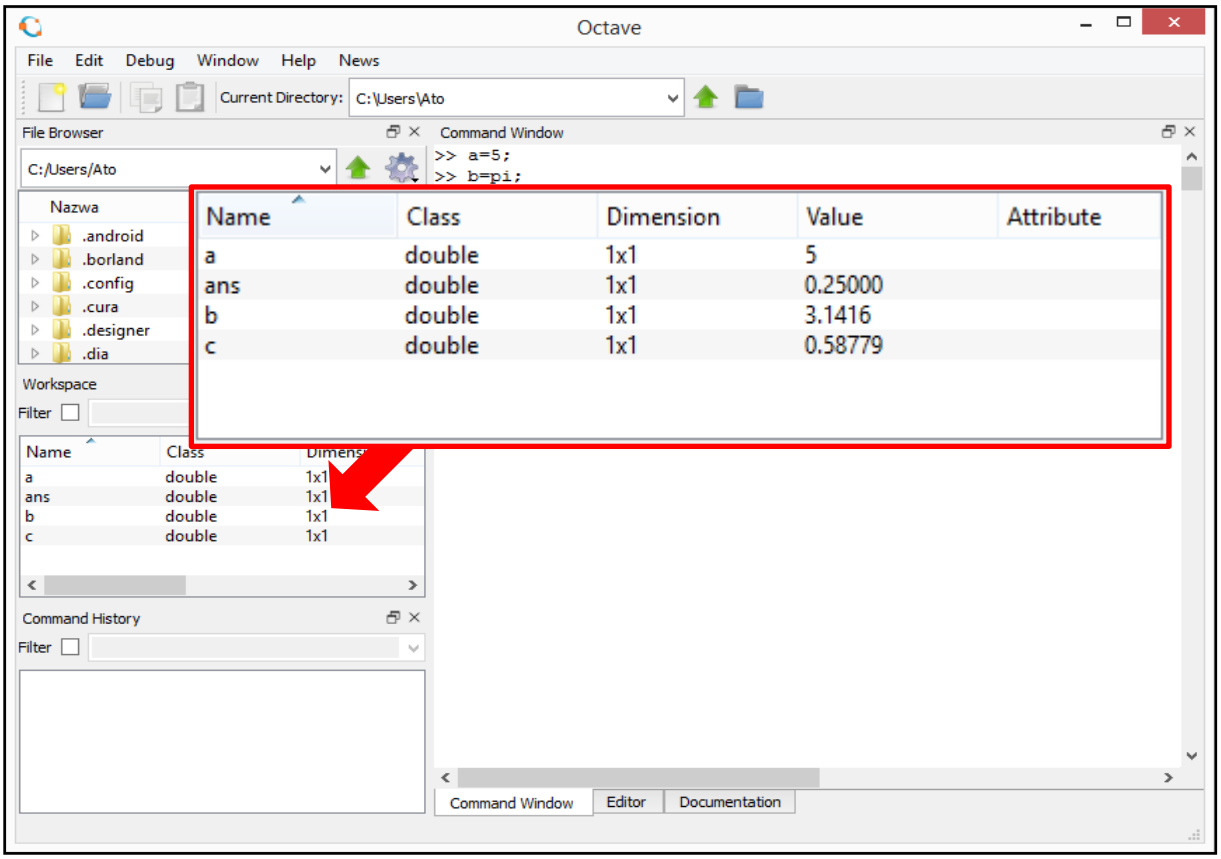

Fig. 39 Finestra "Workspace"

3.4. Editor

Come abbiamo detto prima, possiamo inserire comandi anche usando l’editor. In questo caso, dovete prima scrivere tutti i comandi e poi cliccare su "esegui" lo script. Andate su "Editor" (Fig. 33) e inserite i comandi (il cosiddetto codice). Prima di tutto, comunque, è utile impostare la cartella nella quale si vuole lavorare. Per fare ciò, cliccate sull’icona blu come nella Fig. 40 e poi selezionate la cartella desiderata (potete, per esempio, creare e scegliere la cartella C:\ HOLOMAKERS\Octave).

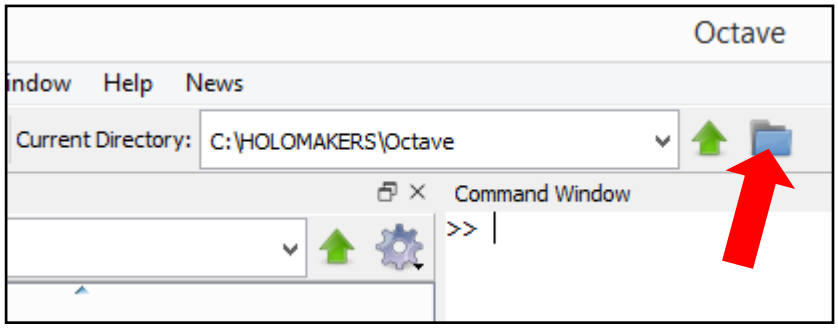

Fig. 40 Cambiare la cartella di lavoro

Adesso possiamo iniziare a scrivere il nostro codice. La Fig. 41 illustra alcuni semplici comandi. Ogni comando appare su una riga diversa (questo perché il codice risulti leggibile e per evitare possibili errori). Per iniziare a calcolare, è necessario eseguire il codice con il pulsante "Salva file ed esegui" segnalato nella Fig. 41 con una freccia rossa. La prima volta il programma ci chiederà di salvare il file, che dovrà essere aggiunto alla raccolta precedente, Current Directory. Nominiamo il file "test01". Il file andrà salvato come "test01.m", e poi il codice verrà eseguito (si eseguiranno tutti i comandi in ordine dalla prima all'ultima riga).

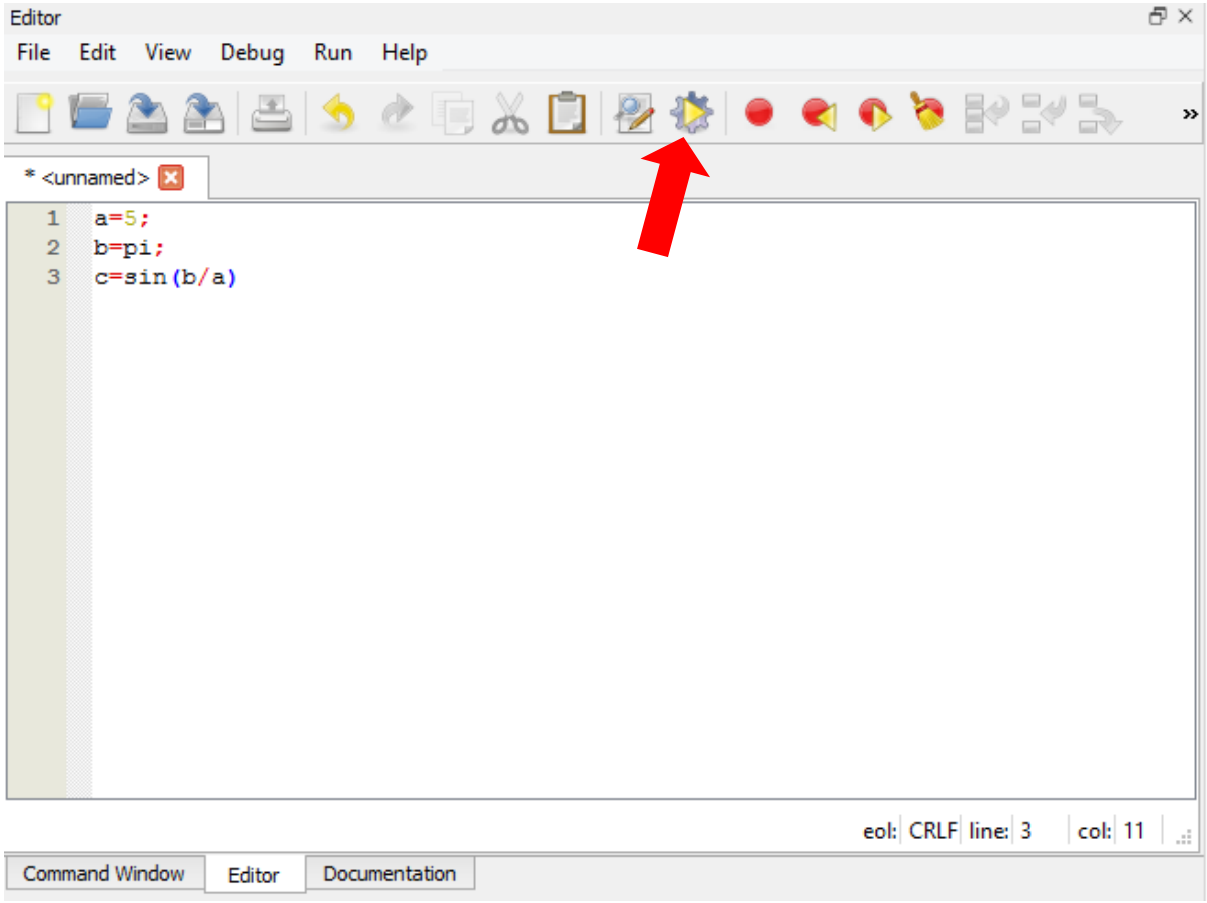

Fig. 41 Editor

Per controllare i risultati delle nostre operazioni, dobbiamo tornare su "Command Window" (Fig. 42). Solo la variabile "c" era visualizzabile, perché il comando c = sin (b/a) non terminava con punto e virgola. I primi due comandi vengono fatti terminare con punto e virgola quindi non compaiono.

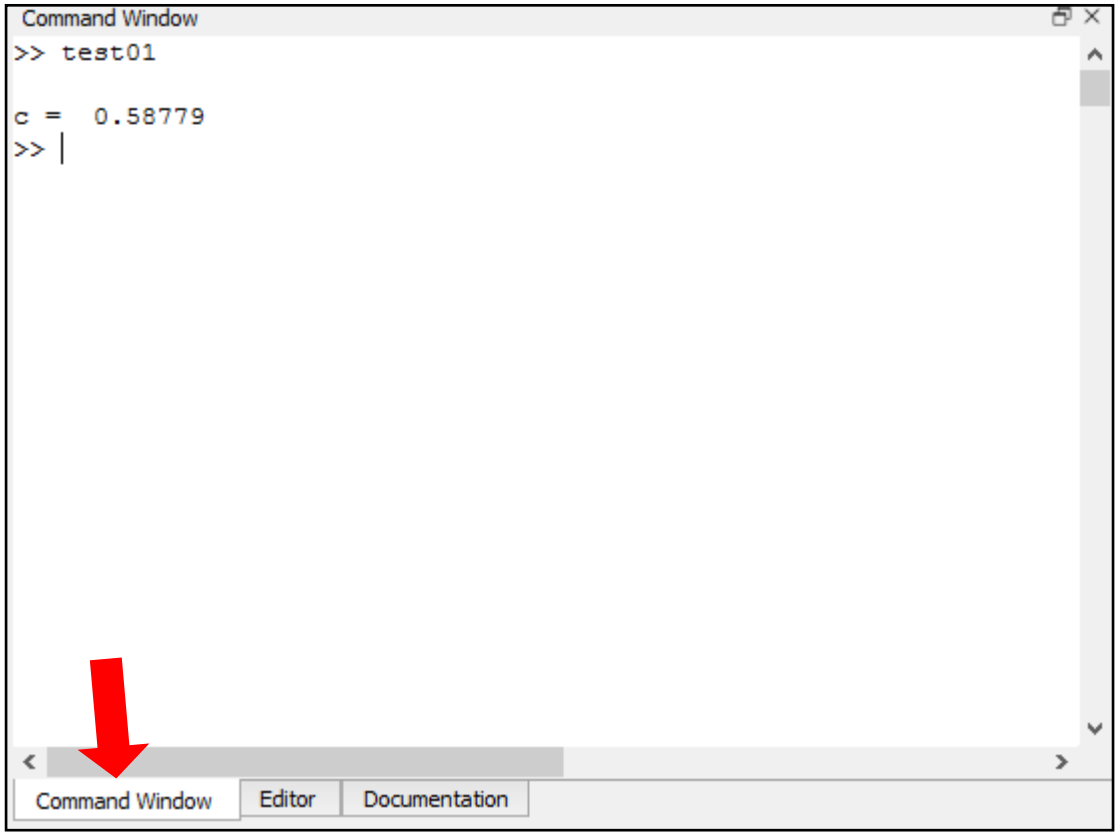

Fig. 42 Risultato del calcolo

3.5. Questioni principali di programmazione su Octave

3.5.1. Variabili

Le variabili vengono usate per conservare i dati che ci servono (numeri, caratteri, testo) nella memoria del computer. Grazie a questa funzione, possiamo vedere il valore della variabile in ogni momento. In Octave, impostare una variabile è molto facile. Scrivete il suo nome e assegnatele un valore, come abbiamo visto nell'ultimo capitolo. In Octave, una variabile può contenere molti numeri. Tale variabile sarà chiamata matrice. Le matrici possono essere a una dimensione o a due dimensioni, come illustrato nella Fig. 43.

matrice a una dimensione

a(1)	a(2)	a(3)	...	a(n)

matrice a due dimensioni

a(1,1)	a(1,2)	a(1,3)	...	a(1,n)
a(2,1)	a(2,2)	a(2,3)	...	a(2,n)
a(3,1)	a(3,2)	a(3,3)	...	a(3,n)
...	...	...	...	...
a(m,1)	a(m,2)	a(m,3)	...	a(m,n)

Fig. 43 Struttura dati della matrice

Come potete vedere, ogni elemento di una matrice a una dimensione è definito da un indice e nel caso della matrice a due dimensioni - da due indici. Immaginate di voler impostare una variabile con il nome tab1, e di attribuirgli 5 numeri. In Octave, scriverete:

```
tab1 = zeros(5,1)
```

È una tabella con 5 righe e una colonna. Ognuno dei cinque elementi della matrice ha valore 0. Adesso impostiamo una matrice a due dimensioni 3x3 con tutti i valori zero:

```
tab2 = zeros(3,3)
```

Inserite le precedenti istruzioni nella "Command Window" per vedere le matrici che avete creato.
Quello che segue è un esempio interessante della creazione di una matrice a una dimensione che consiste in una serie di numeri equidistanti da un certo range. Creiamo una matrice X, che contenga 5 elementi in un range da 0 a π. Useremo la funzione linspace (inizio range, fine range, numero di elementi):

```
X=linspace(0,pi,5)
```

Inserite in Octave l'istruzione precedente nella "Command Window" per vedere il risultato.

3.5.2. Operazioni input/output di base

L'operazione di input di cui parleremo consiste nell'inserire dati con la tastiera (l'utente inserisce con la tastiera dei valori che saranno poi memorizzati dal programma nella variabile appropriata). Un'operazione di output è, ad esempio, quella di far apparire il risultato del calcolo sullo schermo.
Per leggere un numero da tastiera e salvarlo come variabile "a", usate il comando seguente: `a=input('User text')`. Digitate questo comando nella Command Window e poi confermate cliccando Invio.

```
a=input('Enter a value: ')
```

La riga seguente farà comparire il testo: `Enter a value:` e il programma attenderà che inseriate un valore dalla tastiera. Digitate il numero 5 e date conferma col tasto Invio Da questo momento, la variabile "a" contiene il valore 5. Dovreste ottenere un risultato come nella Fig. 44.

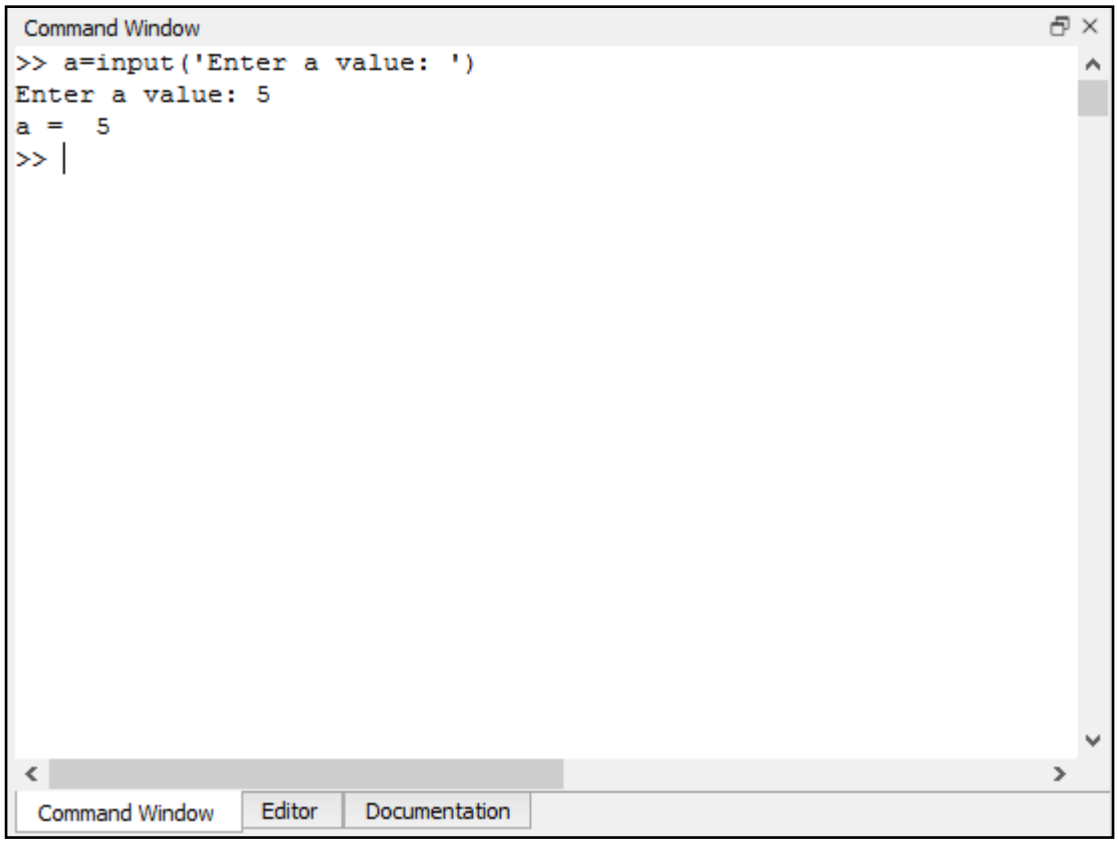

Fig. 44 Inserimento dati da tastiera

La più semplice operazione di output consiste nel far comparire un testo sullo schermo con il comando `disp`. Nella "Command Window" inserite il comando seguente e confermate con Invio:

```
disp('The variable "a" contains the value: '), disp(a)
```

Comparirà il seguente testo: `The variable "a" contains the value:` e nella riga successiva il valore della variabile "a". Quindi, usando la funzione `disp` è possibile far comparire sia il testo che i valori della variabile sullo schermo. Se invece vogliamo far comparire solo il testo, possiamo anche usare la funzione `puts`.

3.5.3. Operatori

Parleremo di tre classi di operatori: aritmetici, relazionali e logici.

1. Gli operatori aritmetici non sono altro che somma (+), sottrazione (-), moltiplicazione (*), divisione (/) e assegnazione dei valori (=). Quindi, per esempio, l'espressione a = c + 2 * b contiene tre operatori (=, +, *).
2. Gli operatori relazionali sono usati per mettere in comparazione due valori (ad esempio, a>b). Questi operatori sono: uguale a (==), minore di (<), maggiore di (>), minore o uguale a

(<=), maggiore o uguale a (> =), diverso da (! =). Il risultato della comparazione tra due valori è il cosiddetto valore logico, che può essere vero o falso. Ad esempio, se scriviamo l'espressione 2 == 5, sarà falso, mentre quello dell'espressione 2 <5 sarà vero.
3. Gli operatori logici, come suggerisce il nome, sono usati per eseguire le operazioni logiche di base come la somma logica (||), la congiunzione logica (&&) e la negazione (!). Il risultato di un'operazione logica è vero o falso. Ad esempio, l'espressione (5> 2) && (2 <3) è vera perché i termini 5> 2 e 2 <3 sono entrambi veri. Un'espressione utilizzata in questo contesto contiene sia gli operatori relazionali (<,>) che logici(&&).
Un concetto importante è la cosiddetta priorità degli operatori. Benché possa sembrare oscuro, ogni studente ci si è confrontato durante gli anni di studio della matematica. Pensate al risultato della seguente espressione: 5 + 2 * 3 = ? Esatto, il risultato è 11. Noterete che, intuitivamente, avete prima moltiplicato e poi sommato. Perché non avete prima sommato 5 e 2 e poi moltiplicato il risultato per 3? Perché siete consapevoli che la moltiplicazione si esegue prima dell'addizione. Questa è la priorità degli operatori. Ci dice quali operazioni all'interno di un'espressione vengono eseguite prima e quali dopo. La tabella 1 illustra l'ordine degli operatori in base alla loro priorità. L'operatore di negazione logica presenta la priorità maggiore. Questo significa che se tale operatore compare da qualche parte nell'espressione andrà eseguito per primo. Gli operatori relazionali hanno la priorità minore (vengono eseguiti solo alla fine).

Priority	Operator
1	!
2	&&, *, /
3	||, +, -
4	==, <, >, <=, >=, !=

Table 1 Priorità degli operatori

Un'altra questione ovvia che abbiamo imparato durante le lezioni di matematica è il fatto che se vogliamo invertire l'ordine in cui le operazioni vengono eseguite, dobbiamo mettere le parentesi al posto giusto. Dunque, l'espressione 5 + 2 * 3 conferisce il valore di 11 mentre il risultato di (5 + 2) * 3 sarà invece 21. Talvolta è necessario inserire delle parentesi perché l'espressione abbia senso. Questo è l'esempio di prima: (5> 2) && (2 <3).

3.5.4. Proposizione condizionale

Ognuno di noi prende delle decisioni tutti i giorni. Per esempio, se c'è il sole, possiamo decidere di uscire in bicicletta, in caso contrario rimaniamo a casa. Questo è un esempio di proposizione condizionale semplice. Innanzitutto, valutiamo le condizioni (ad esempio se c'è il sole) e poi, in base al risultato della valutazione, ci comportiamo di conseguenza (ad esempio usciamo in bicicletta oppure rimaniamo a casa). La condizione è rappresentata da un'espressione logica che è vera o falsa. L'espressione logica può essere ad esempio la frase:

“Adesso il tempo è sereno." Questa frase (espressione logica) è vera o falsa a seconda delle condizioni meteorologiche.
Nell’ambito della programmazione, le proposizioni condizionali sono necessarie quanto nella vita di tutti i giorni. La Fig. 45 illustra la struttura di una frase condizionale semplice. In base al fatto che la condizione sia vera oppure falsa, verranno eseguite le altre istruzioni (proposizioni).

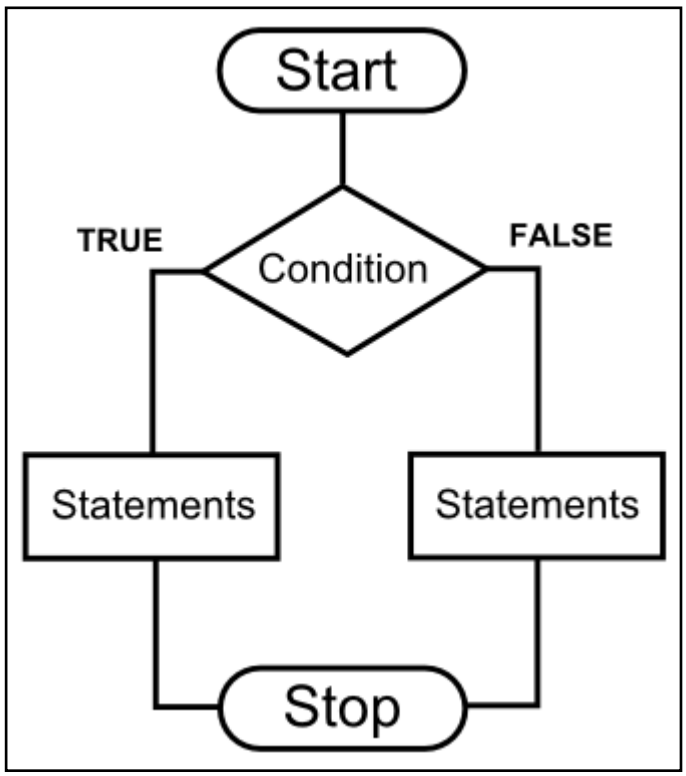

Fig. 45 Algoritmo di una proposizione condizionale semplice

In Octave, le proposizioni condizionali semplici hanno questo aspetto:

```
if (condition)
```
Altre proporzioni che verranno eseguite se la condizione è vera
```
else
```
Altre proporzioni che verranno eseguite se la condizione è falsa
```
endif
```

Come esempio di utilizzo di una proposizione condizionale, scriviamo un breve codice che controllerà se il numero inserito dall'utente sia negativo o non-negativo. Scrivete nella finestra dell’Editor il codice come nella Fig. 46. Poi cliccate su "salva file e esegui"  e aprite la Command Window. Dopo aver inserito il numero e premuto Invio, comparirà la frase corrispondente.

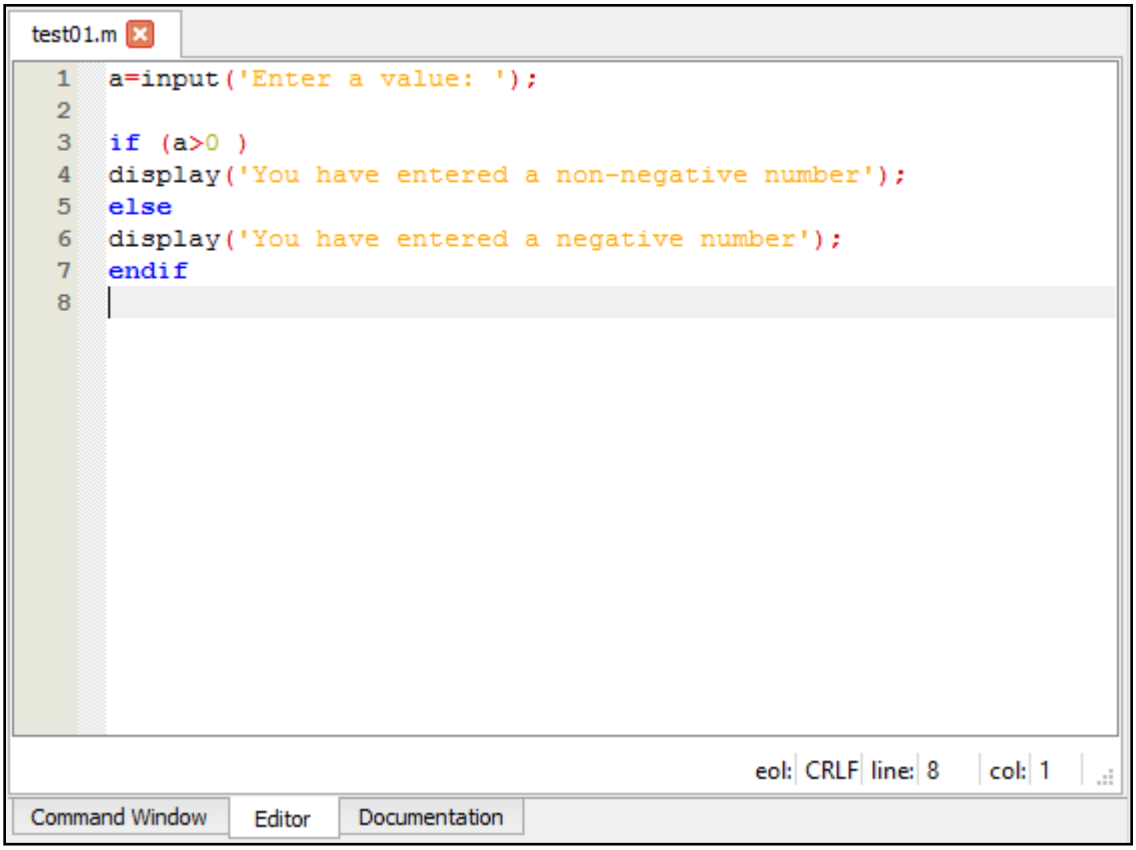

Fig. 46 Esempio di utilizzo di una proposizione condizionale

Spesso, tuttavia, ci troviamo nella situazione in cui, dopo aver verificato una condizione, ne vogliamo verificare un'altra. In Octave, quindi, dobbiamo scrivere:

`if (condition1)`
Altre proposizioni che verranno eseguite se `condition1` è vera
`elseif (condition2)`
Altre proposizioni che verranno eseguite se `condition1` è falsa e `condition2` è vera
`else`
Altre proposizioni che verranno eseguite se sia `condition1` che `condition2` sono false
`endif`

Per spiegarlo meglio, modifichiamo leggermente l'ultimo codice in modo che il programma ci indichi anche se l'utente ha inserito il numero zero (Fig. 47).

```
1 a=input('Enter a value: ');
2
3 if (a>0 )
4 display('You have entered a positive number');
5 elseif (a<0)
6 display('You have entered a negative number');
7 else
8 display('You have entered 0');
9 endif
```

Fig. 47 Esempio di utilizzo della proposizione condizionale

3.5.5. Loop "per"

Spesso si presenta la necessità di eseguire più volte gli stessi comandi. Per evitare di inserire lo stesso comando molte volte, il programma utilizza il loop. Parleremo brevemente di un solo tipo di loop – il loop "per", che permette di eseguire una serie di comandi un certo numero di volte. La struttura del loop per viene mostrata nella Fig. 48. È molto semplice e ci dice che le proposizioni compariranno, in questo caso, dieci volte. Come sappiamo che saranno proprio 10 volte? Questa informazione viene definita all'inizio del loop. Una variabile, che abbiamo chiamato "i" in ogni loop seguente, aumenta il suo valore di 1, da 1 a 10. Quando la variabile "i" raggiunge il massimo del valore (10, in questo caso), il loop termina l'operazione.

```
1  for i=1:10
2
3  statements...
4
5  end
```

Fig. 48 Loop "per"

Un esempio molto semplice di utilizzo del loop "per" è scrivere un numero di caratteri sullo schermo. Immaginate che vogliamo far comparire una riga con i caratteri "-". A questo scopo possiamo scrivere un codice molto semplice, come mostrato nella Fig. 49. Create un nuovo codice cliccando sull'icona  . Per la prima esecuzione, salvatelo con il nome di test02. Potrete vedere che la nostra riga sarà di 15 caratteri (il loop "per" verrà eseguito 15 volte).

```
1  for i=1:15
2
3    puts('-');
4
5    end
```

Fig. 49 Visualizzare una riga di 15 caratteri

Il risultato della scrittura del codice può essere visualizzato nella Command Window (Fig. 50).

```
Command Window
>> test02

--------------->>
```

Fig. 50 Una riga di 15 caratteri

Possiamo modificare leggermente il codice per permettere all'utente di stabilire la lunghezza della riga da visualizzare (Fig. 51). Vi preghiamo di notare che il loop verrà eseguito n volte, dove n è il numero impostato dall'utente.

```
n=input ('enter the length of the line: ');

for i=1:n

puts('-');

end
```

Fig. 51 Visualizzare una riga con qualsiasi numero di caratteri

Il loop “per" è usato spesso per vari tipi di calcolo. Sapete come si scrive un codice di 5 numeri in modo che poi il programma ne calcoli la media? Se la risposta è no, date un’occhiata alla Fig. 52.

```
for i=1:5

a(i) = input('Enter a value: ');

end

sum = 0;

for i = 1:5

sum = sum + a(i);

end

average = sum/5
```

Fig. 52 Calcolo della media aritmetica

Analizziamo adesso questo codice. Innanzitutto l’utente inserisce 5 numeri dalla tastiera. Questi numeri sono scritti in una matrice "a". Poi imposta la variabile “somma” con valore pari a 0. Il loop successivo esegue la somma dei 5 numeri inseriti dall’utente. L’ultima riga corrisponde al calcolo della media aritmetica. Siccome non termina con un punto e virgola, il risultato comparirà sullo schermo.

3.5.6. Disegnare grafici

Octave permette di disegnare il grafico di una funzione in modo molto semplice. Per disegnare una funzione $y = f(x)$ dobbiamo prima creare la matrice degli elementi x e la corrispondente matrice dei valori $f(x)$. Supponete di voler disegnare la funzione $y = \sin(x)$ in un range di $[0,2\pi]$. Octave permette di impostare una matrice con n elementi equidistanti dal range $[a,b]$. Ciò si può fare usando la funzione `linspace(a, b, n)`. Assumiamo di voler dividere il nostro range $[0, 4\pi]$ in 100 valori equidistanti. In questo caso dobbiamo scrivere:

```
x = linspace(0, 4*pi,100);
```

Così abbiamo impostato la matrice di 100 elementi. Ottenere una matrice di valori di funzione è ancora più semplice, poiché basterà scrivere:

```
y=sin(x);
```

Dato che x è una matrice a 100 elementi, allora y diventerà automaticamente una matrice con lo stesso numero di elementi. Adesso visualizziamo il grafico. Per farlo, scriviamo:

```
plot(x, y)
```

L'intero codice è riportato nella Fig. 53.

```
1  x = linspace(0, 4*pi,100);
2  y = sin(x);
3  plot(x, y)
```

Fig. 53 Preparazione dei dati e visualizzazione del grafico

Dopo aver iniziato il codice apparirà una nuova finestra con il grafico (vedi Fig. 54).

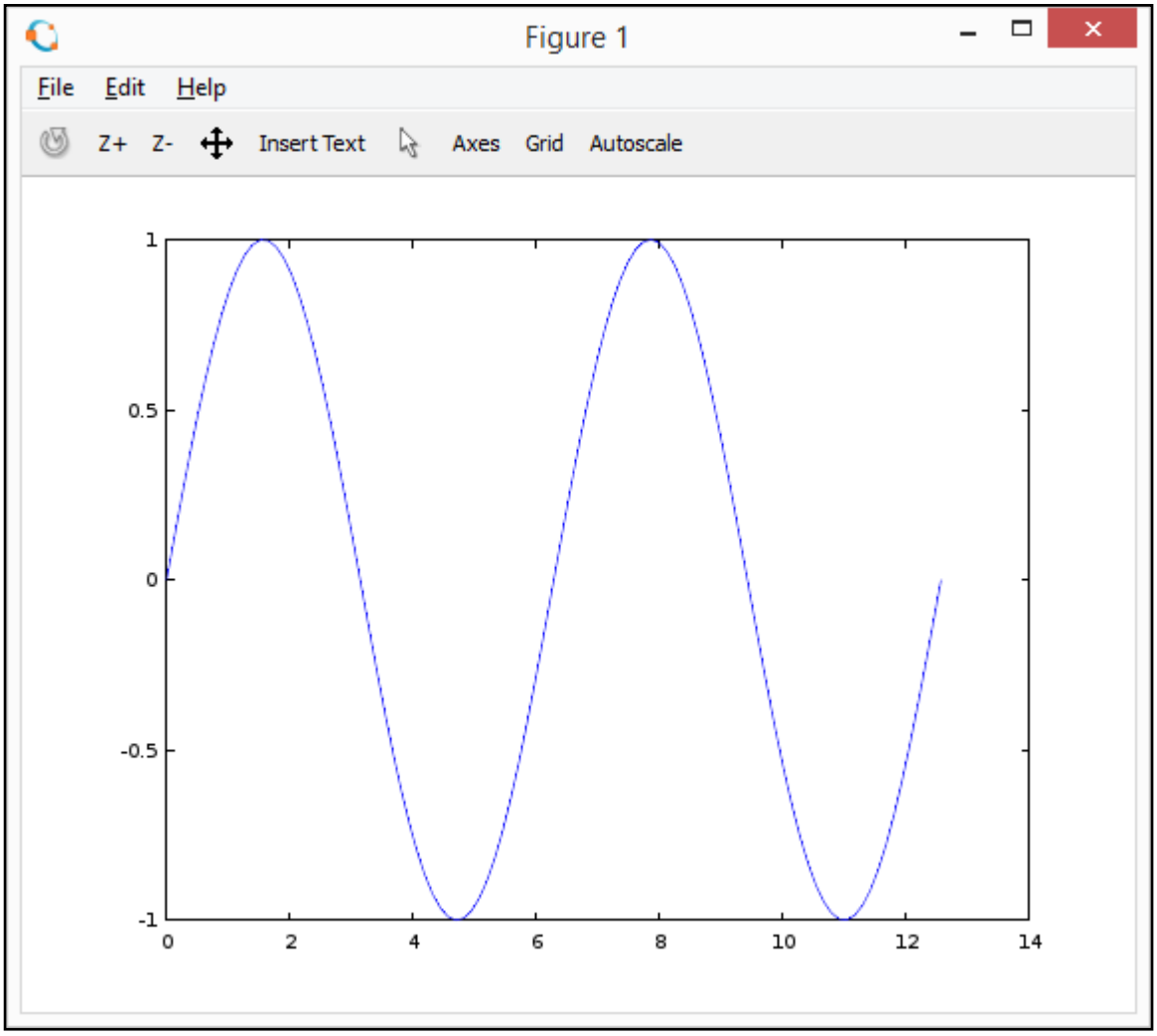

Fig. 54 Grafico

3.6. Gestione delle immagini

3.6.1. Creare un'immagine

Lei immagini grigie, in Octave, sono rappresentate come matrici a due dimensioni, in cui ogni elemento corrisponde a un pixel (Fig. 55).

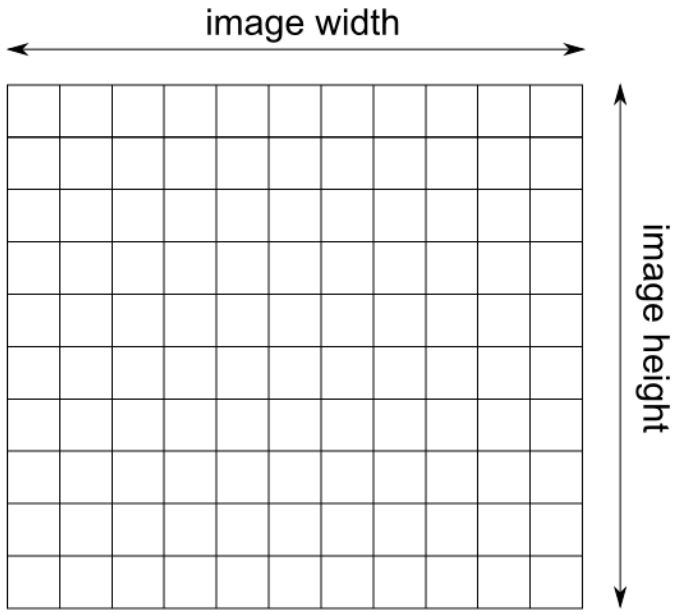

Fig. 55

Creare un'immagine consiste semplicemente nell'impostare una matrice a due dimensioni. Se volete creare un'immagine che misura 200x200 pixel, scrivete:

```
image1 = zeros(200,200);
```

In questo modo, la variabile `image1` è impostata come matrice 200x200 pieno di zeri (ogni elemento della matrice è zero). Abbiamo usato la funzione `zeros`.

3.6.2. Leggere/scrivere un'immagine e visualizzarla sullo schermo

Spesso c'è la necessità di caricare il file di un'immagine già presente sul disco. Per fare ciò, copiate innanzitutto l'immagine nella cartella precedente "current directory". Poi scrivete il comando:

```
image1 = imread('filename');
```

Il file dell'immagine con il nome specificato adesso è conservato nella variabile image1.
Per visualizzare l'immagine dalla variabile image1, usate il comando:

```
imshow(image1);
```

Per visualizzare più immagini, a ognuna dovrebbe essere assegnato un numero diverso con il comando `figure(number)`. È anche utile nominare la finestra con il comando `title('Image Title')`. Vedete qui sotto un esempio di visualizzazione di due immagini in in due diverse finestre:

```
figure(1);imshow(image1);title('Image no 1');
figure(2);imshow(image2);title('Image no 2');
```

L'immagine viene salvata in un file con il comando:

```
imwrite(image1,'name.extension');
```

L'immagine può essere salvata in vari formati, come bmp, png, jpg, tiff, gif.
Come esempio, caricate un'immagine con estensione bmp sull'attuale cartella di lavoro e poi scrivete un codice che farà apparire questa immagine sullo schermo, e salvatela con diverse estensioni:

```
1  image1=imread('test.bmp');
2  imshow(image1);
3  imwrite(image1,'test.gif');
4  imwrite(image1,'test.tiff');
5  imwrite(image1,'test.png');
6  imwrite(image1,'test.jpg');
```

Fig. 56 Convertire l'immagine in formati differenti

3.6.3. Conversione del colore

Octave offre molte funzioni differenti per la conversione del colore. Tra queste ce n'è una che converte il colore dell'immagine in grigio e una che converte un'immagine (grigia o colorata) in un'immagine binaria composta da due soli colori – bianco e nero. Queste funzioni, tuttavia, non sono disponibili di default col programma Octave. Per poterle usare è necessario scaricare il pacchetto `image`. Quindi, all'inizio del codice, scrivete:

```
pkg install image
pkg load image
```

Da questo momento possiamo usare liberamente le molte funzioni incluse nel pacchetto. Per convertire il colore di un'immagine in grigio, usate la funzione `rgb2gray`. Per esempio, se la variabile `colorImage` contiene il colore dell'immagine, possiamo sostituirlo col grigio e salvarla come variabile `grayImage` come segue:

```
grayImage= rgb2gray(colorImage);
```

Analogamente, l'immagine può essere convertita in una binaria:

```
bwImage= im2bw(colorImage,0.5);
```

3.6.4. Rotazione

In Octave, è possibile ruotare un'immagine centrale di qualsiasi angolo grazie alla funzione `imrotate`. Per usare questa funzione, dobbiamo indicare l'immagine da ruotare e l'angolo di rotazione. Ad esempio, se volete ruotare di 45° `Image1` scrivete, nella variabile `Image2`:

```
Image2 = imrotate(Image1,45);
```

3.6.5. Addizione, sottrazione, moltiplicazione, divisione.

Come sappiamo, l'immagine in Octave è rappresentata da una matrice bidimensionale in cui le dimensioni corrispondono alla larghezza e all'altezza dell'immagine in pixel. Avendo due immagini con le stesse dimensioni, possiamo eseguire facilmente operazioni come l'addizione o la moltiplicazione delle immagini. Ad esempio, sommare due immagini tra loro significherà semplicemente sommare i valori dei pixel corrispondenti a ognuna delle due (Fig. 57).

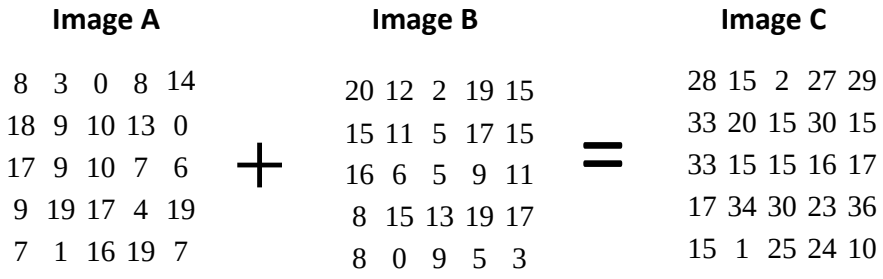

Fig. 57 Adding two images

Allo stesso modo verranno eseguite la sottrazione, la moltiplicazione e la divisione. Nella tabella 2 sono elencati i nomi di particolari funzioni del programma Octave.

Operation	Octave
Somma	`imadd`
Sottrazione	`imsubstract`
Moltiplicazione	`immultiply`
Divisione	`imdivide`

Tabella 2 Operazioni con le immagini

Mettiamo che `imgA` e `imgB` rappresentino due immagini con le stesse dimensioni. Allora potremo creare un'immagine imgC che corrisponderà alla somma o alla moltiplicazione di queste due immagini:

```
imgC = imadd(imgA,imgB);
imgC = immultiply(imgA,imgB);
```

3.6.6. Trasformata di Fourier

La Trasformata di Fourier è un genere di trasformata molto usata in diversi campi di studio come l'elettronica, l'ottica e la musica. E possiamo applicarla anche nel caso della gestione delle immagini. La trasformata di Fourier di un'immagine può assomigliare a una serie di pixel casuali, però contiene le stesse informazioni dell'immagine originale (see Fig. 58).

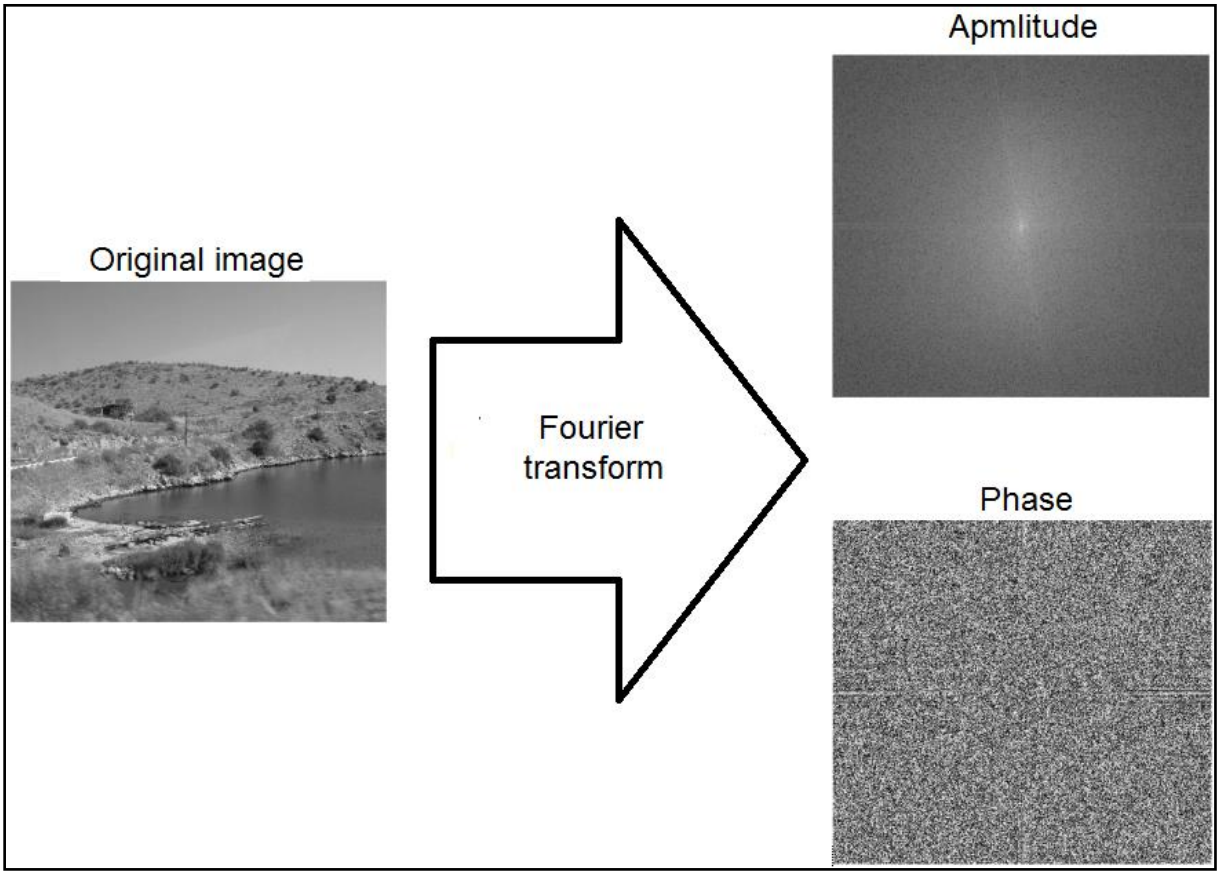

Fig. 58 L'immagine e la sua trasformata di Fourier

Ogni punto della trasformata di Fourier non ha solo un'ampiezza ma anche una fase. Una discussione dettagliata di tale questione va ben oltre i propositi di questa guida. Ricordiamoci solo che la fase è necessaria a riprodurre l'immagine originale (attraverso la trasformata di Fourier inversa), mentre l'ampiezza è distribuita nel modo seguente: i punti al centro della trasformata corrispondono alle aree regolari dell'immagine, mentre i punti distanti dal centro corrispondono ai bordi dell'immagine (un bordo ad esempio è il confine tra il cielo e le montagne, ecc.) Tutto ciò può essere illustrato attraverso un semplice esempio. La Fig. 59a mostra il quadrato bianco e la sua trasformata di Fourier. Il punto in mezzo della trasformata corrisponde alla parte centrale del quadrato (un'immagine definita) mentre ognuno dei quattro "bracci" coincide col lato del quadrato corrispondente (Fig. 59b).

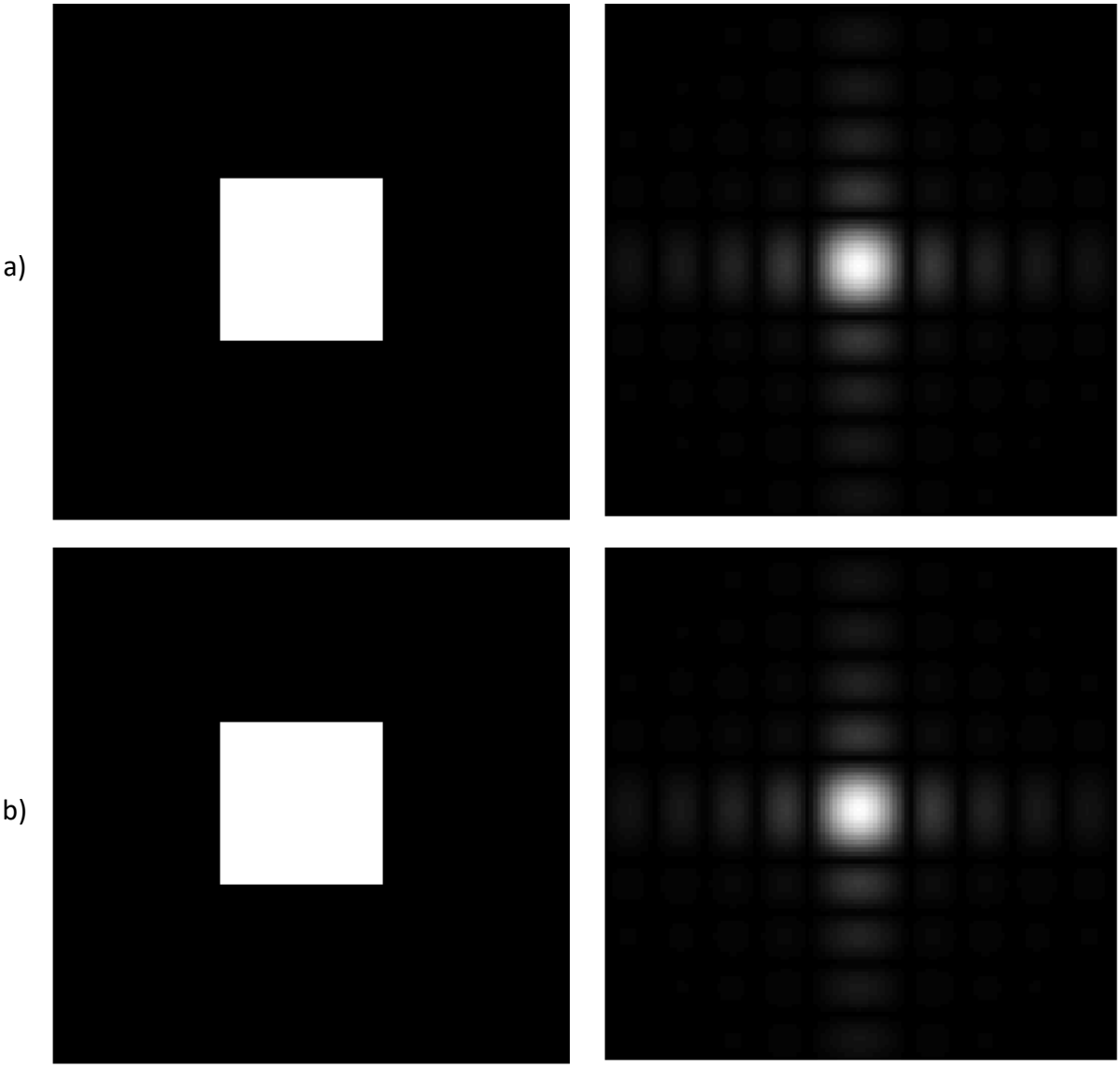

Fig. 59 Trasformata di Fourier del quadrato

Quando si tratta di figure geometriche semplici, è visibile abbastanza chiaramente. La Fig. 60 mostra l'immagine del triangolo e la sua trasformazione. Il punto in mezzo corrisponde alla parte centrale del triangolo. I tre lati del triangolo, nella trasformata, possono essere indicati come i tre "bracci". Sono inoltre visibili altri tre "bracci", che rappresentano i vertici del triangolo.

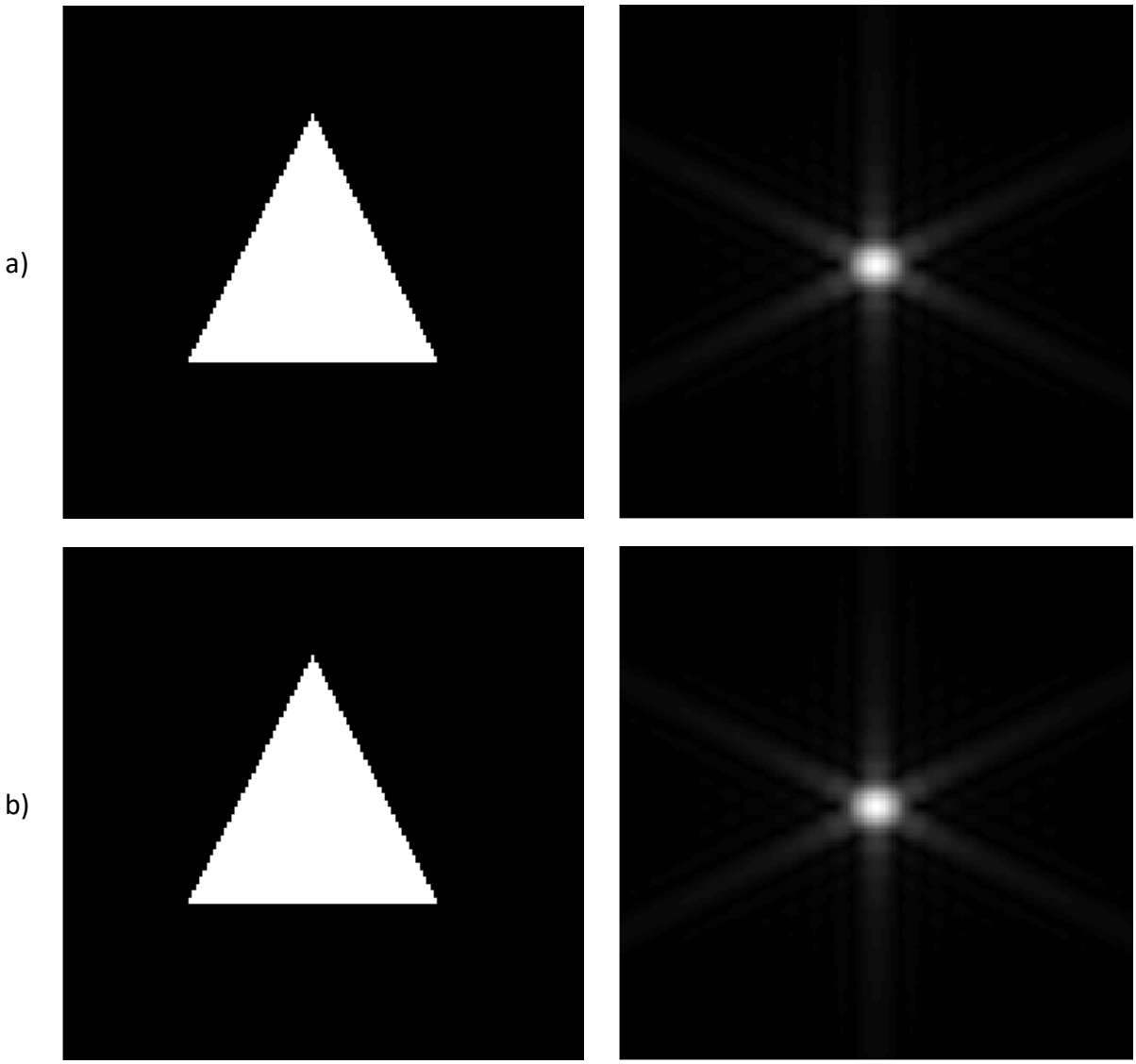

Fig. 60 Trasformata di Fourier del triangolo

In Octave, la funzione della trasformata di Fourier si chiama `fft2`. La funzione si comporta in modo che la trasformata sia spostata dal centro. È quindi necessario usare contemporaneamente la funzione `fftshift`. Ad esempio, se volete applicare una trasformata di Fourier all'immagine `img1`, scrivete:

```
img2=fftshift(fft2(img1));
```

Per visualizzare l'ampiezza della trasformata, scrivete:

```
imshow(abs(img2),[]);
```

Invece, per visualizzare la fase della trasformata:

```
imshow(angle(img2),[]);
```

3.7. Algoritmo per calcolare ologrammi generati a computer (CGH)

Per creare un ologramma analogico dobbiamo disporre di un oggetto reale, che verrà poi registrato come ologramma. Quando si tratta di ologrammi generati a computer, l'oggetto reale è sostituito da un file grafico contenente qualunque immagine (ad esempio un testo bianco su sfondo nero). La generazione di un ologramma implica il calcolo a computer del pattern di interferenza. Il pattern può poi essere registrato su una lastra o pellicola olografica. Dopo aver attraversato l'ologramma, la luce del laser sfumerà in modo da creare un pattern progettato precedentemente (ossia il testo). La generazione di ologrammi a computer viene realizzata tramite l'impiego di un certo algoritmo, l'algoritmo di Gerchberg e Saxton. L'algoritmo richiede che sia impostato l'input di distribuzione dell'intensità desiderata (vale a dire del file grafico con il testo menzionato sopra) e l'intensità di distribuzione del raggio laser in cui l'ologramma verrà riprodotto (anche qui si tratta di un file grafico). Come risultato dell'applicazione dell'algoritmo, è possibile ottenere un file grafico con una distribuzione di fase che coincide con l'ologramma progettato. L'idea generale è illustrata nella Fig. 61.

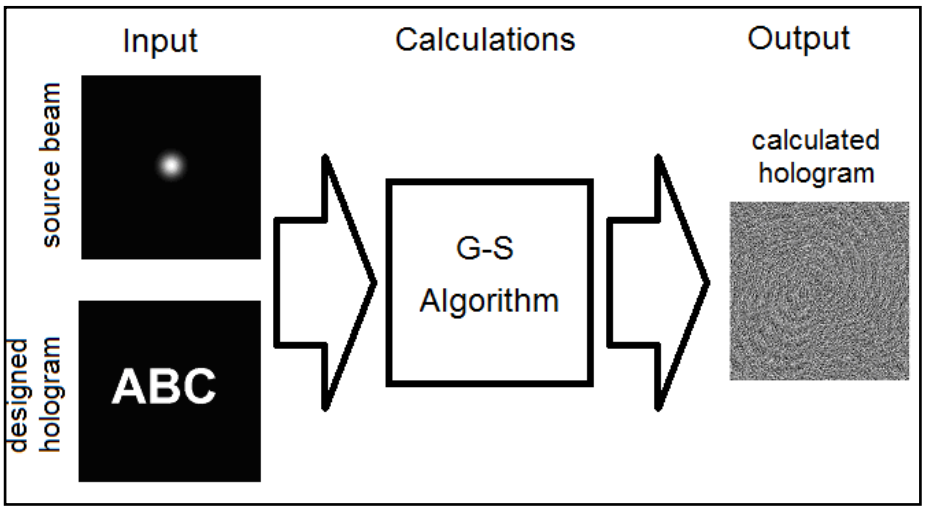

Fig. 61 I dati di input sulla base dei quali viene calcolato l'ologramma

L'algoritmo di Gerchberg e Saxton è un algoritmo iterativo, il che significa che alcune operazioni (calcoli) vengono eseguite diverse volte. La qualità dell'ologramma migliora ogni volta che si esegue il calcolo. Lo schema dell'algoritmo G-S è illustrato nella Fig. 62.

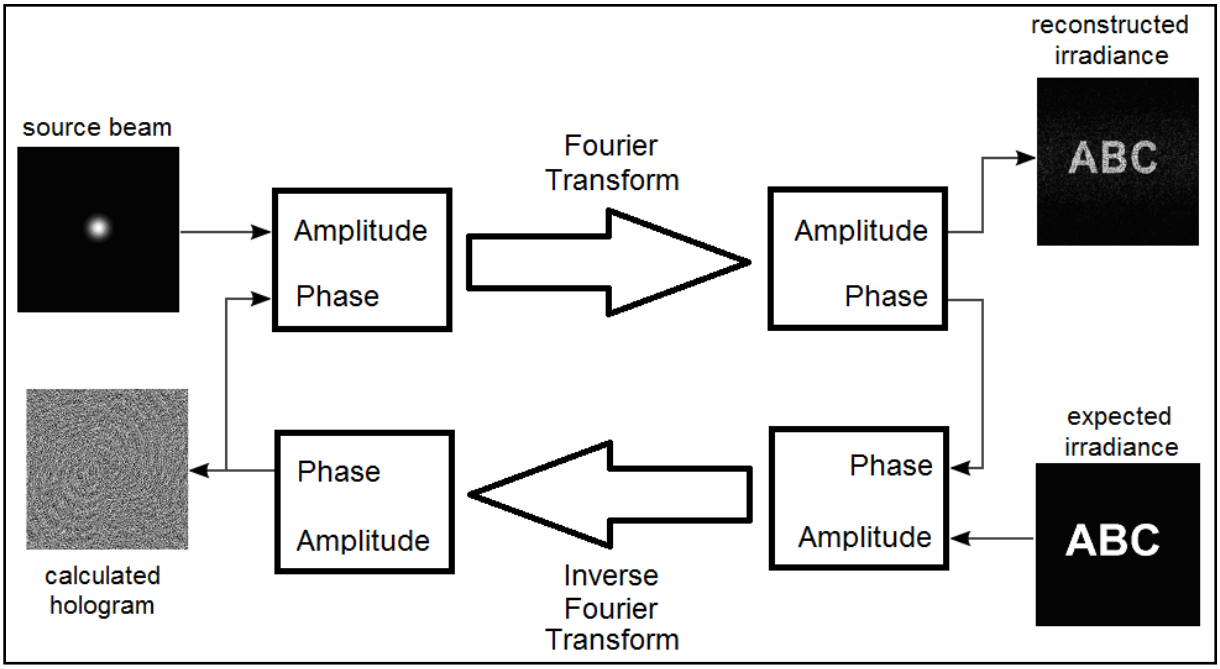

Fig. 62 L'algoritmo di Gerchberg e Saxton

Quello che segue è il codice che useremo per generare ologrammi. L'ologramma che calcoleremo dovrà essere salvato nel file CGH.bmp. I file campione si possono scaricare dal sito holomakers.eu.

```
pkg install image
pkg load image

GaussianBeam = imread('GaussianBeam.png');
GaussianBeam = rgb2gray(GaussianBeam);

StartPhase=zeros(1024,1024);

Source=fft2(ifft2(GaussianBeam));
StartPhase=fft2(ifft2(StartPhase));
Source = abs(Source).*exp(1i*angle(StartPhase));

TargetImg = imread('target.bmp');
Target=fft2(ifft2(TargetImg));

A = fftshift(ifft2(fftshift(Target)));
for i=1:20
  B = abs(Source).* exp(1i*angle(A));
  C = fftshift(fft2(fftshift(B)));
  D = abs(Target).* exp(1i*angle(C));
  A = fftshift(ifft2(fftshift(D)));
end

figure(1);imshow(GaussianBeam);title('Source Intensity');
```

```
figure(2);imshow(Target);title('Expected Intensity');
figure(3);imshow(angle(A),[]);title('Calculated Hologram');
figure(4);imshow(abs(C),[]);title('Reconstructed Intensity');

A=angle(A);
A=A-min(A(:));
A=A/max(A(:));

imwrite(A, 'CGH.bmp');
```