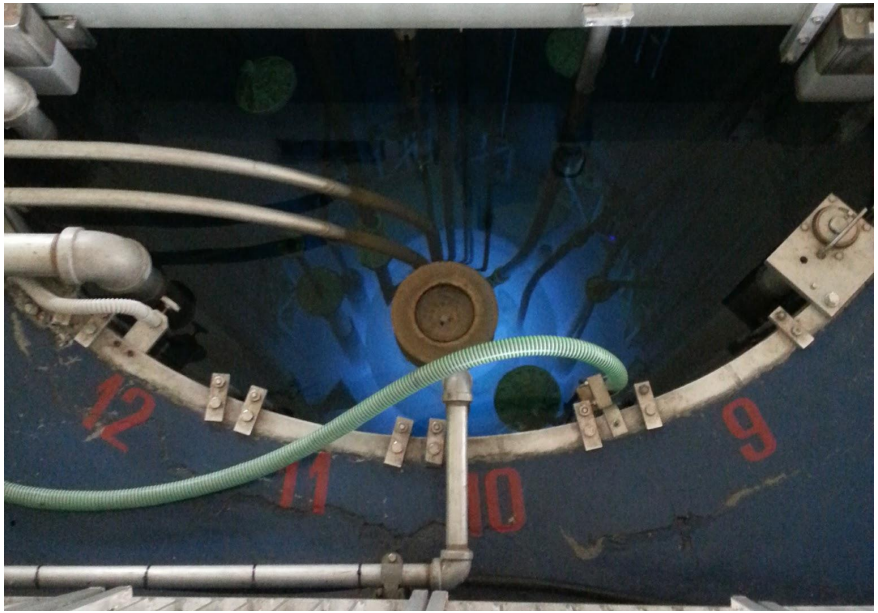
Algoritmo GPU per fit di cerchi di fotoni Cherenkov



15 giugno 2017, Stage estivo del dipartimento di Fisica dell'Università di Pavia.
Luce Cherenkov nel reattore nucleare di ricerca LENA.

Effetto Cherenkov

- **Particelle** che viaggiano **più veloce** della **luce** nel **mezzo** generano fotoni ad un determinato angolo rispetto alla direzione di propagazione.
- Su una superficie perpendicolare a tale direzione, i fotoni si distribuiscono quindi su **circonferenze**. I rivelatori Cherenkov sono composti da griglie di fotodetector ognuno dei quali fornisce un hit.

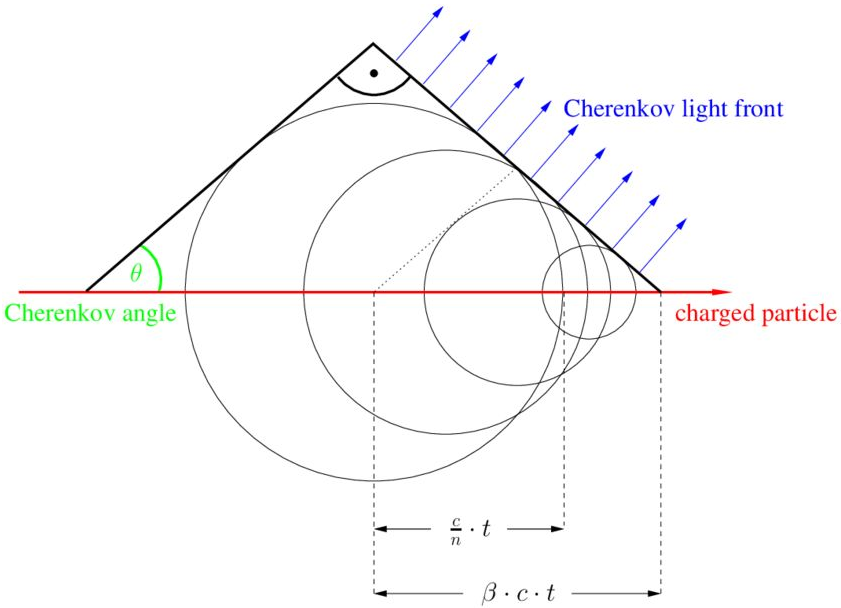


Figura 1: Rappresentazione dell'effetto Cherenkov di una particella con velocità relativa β = v c.

Problema computazionale

La ricostruzione del segnale prevede la **ricostruzione** di **cerchi** a partire da una collezione di hit, che presentano sia **punti mancanti** (i cerchi non sono completi) che **rumore**.

Questo problema è **parallelizzabile** e quindi è possibile affrontarlo tramite l'utilizzo di una **GPU**.

Ci sono 2 possibili approcci che sfruttano la potenza della GPU in modo leggermente diverso:

- L'algoritmo "**Almagest**" basato sul **Teorema** di **Tolomeo**
- Un algoritmo basato su un **Istogramma** delle **Distanze**

Almagest

- Algoritmo **combinatorio**, parallelizzabile assegnando a ogni **thread** una **terna** di hit e procedendo in questo modo:
  - per ogni terna, fare un **check** su tutti i **punti rimanenti**, qualora questi soddisfino il **teorema** di **Tolomeo** sui quadrilateri **ciclici**, vengono presi in considerazione come punti del cerchio, altrimenti vengono esclusi.
  - Una volta **separati** i singoli cerchi, effettuare un **fit** per trovare la posizione del **centro** e il **raggio**.

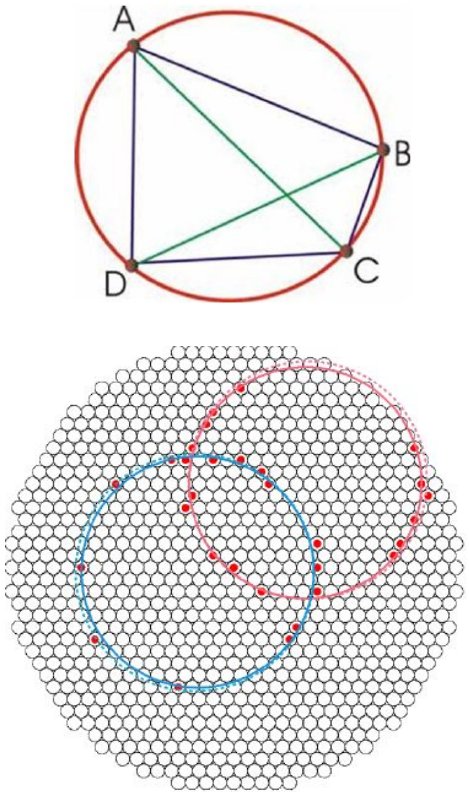


Figura 2
in alto: rappresentazione grafica del teorema di Tolomeo
in basso: esempio di ring fit con questo algoritmo

Istogramma delle distanze

- Divisione del **piano XY** in una **griglia**
- Ad ogni **thread** viene assegnato un **punto** sulla griglia
  - La thread costruisce **l'istogramma** delle **distanze** dal punto a lei assegnato ai vari punti di hit dei rivelatori
  - qualora questo istogramma presenti un **picco** superiore a un **valore prestabilito**:
    - il **punto** sulla **griglia** viene considerato come **centro** di un cerchio
    - il valore corrispondente al picco sull'istogramma corrisponde al **raggio** del cerchio fittato

istogramma delle distanze (esempi realizzati con ROOT)

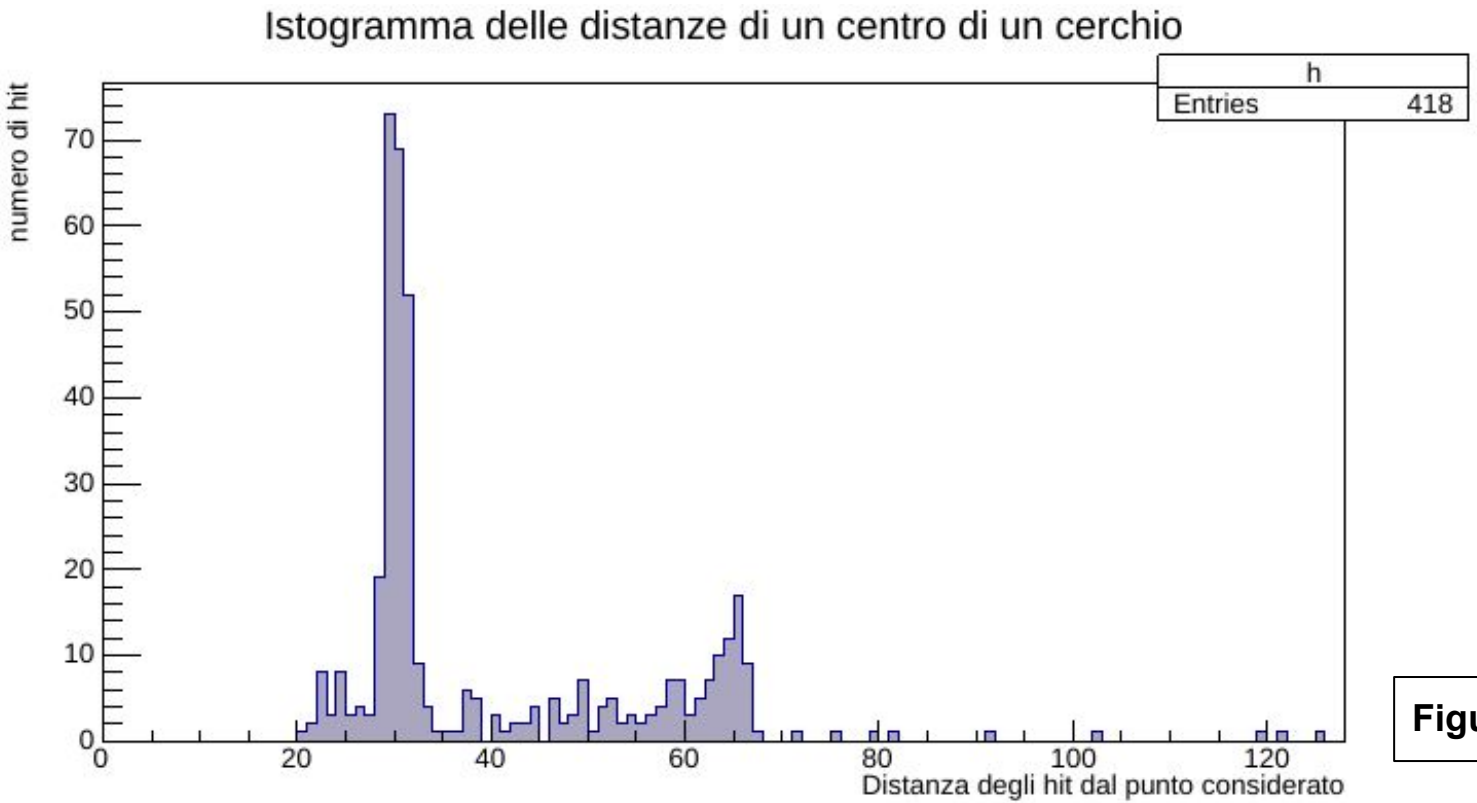


Figura 3

istogramma delle distanze (esempi realizzati con ROOT)

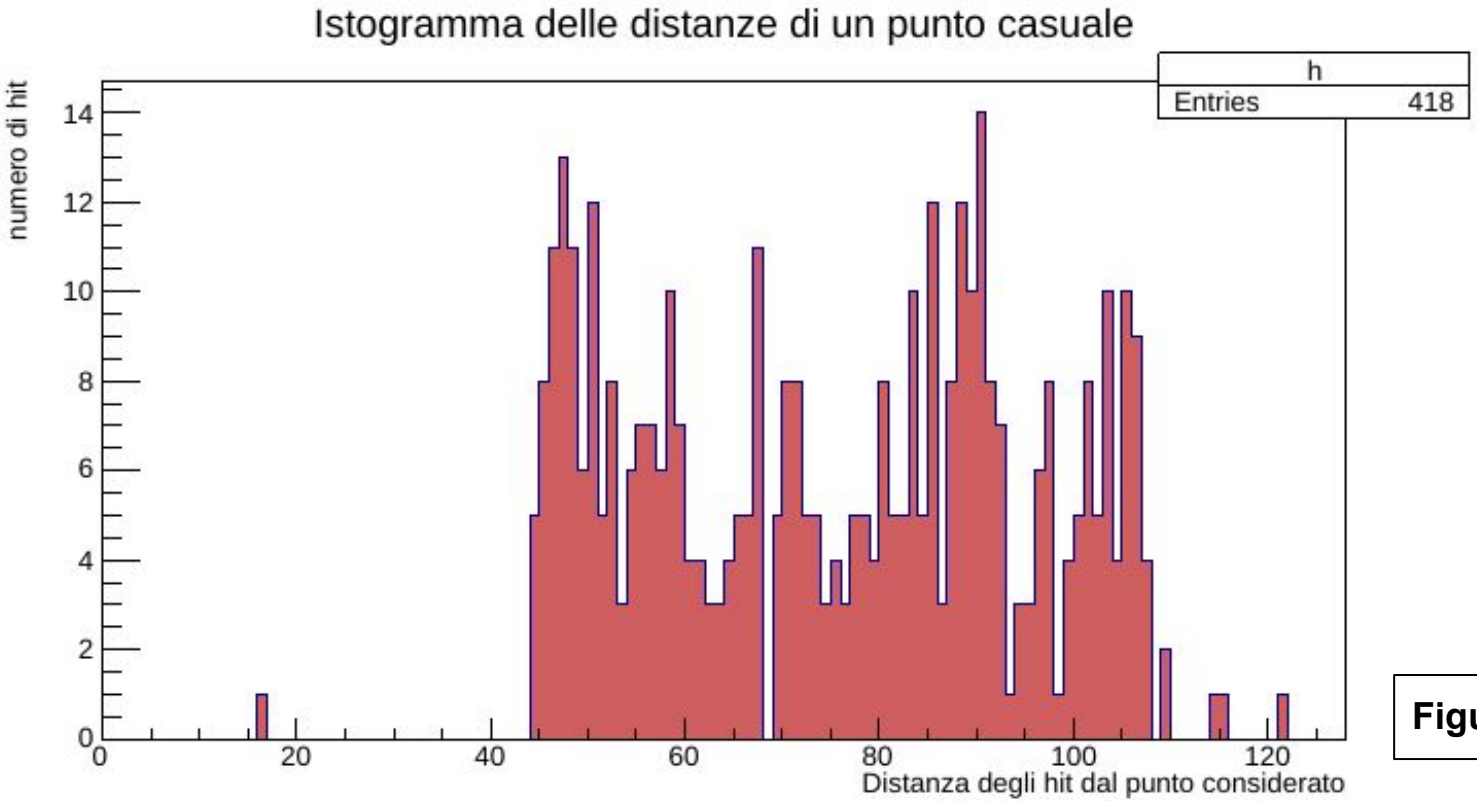


Figura 4

Soluzione scelta

- Realizzazione di una simulazione **Monte Carlo** per la generazione dei dati, ovvero hit distribuiti su **cerchi** di vario **raggio** e **posizione**, con la possibilità di includere del **rumore** nei vari hit.
- Implementazione dell'algoritmo basato **sull'Istogramma** delle **Distanze** per la ricostruzione dei cerchi, sia nel caso di dati di cerchi **singoli** che **multipli**

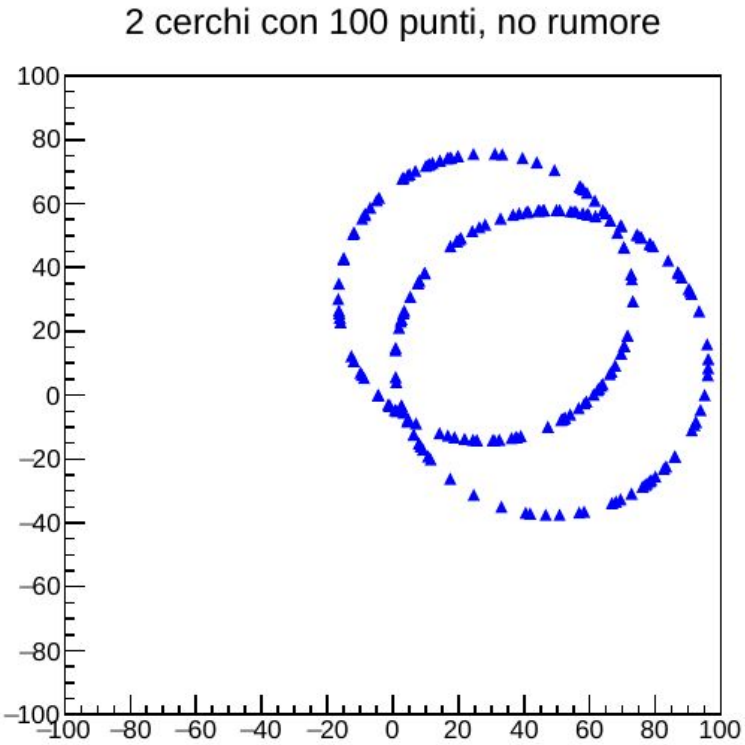
Un possibile sviluppo **futuro** può essere quello di implementare l'algoritmo **Almagest** per dividere i dati nei singoli cerchi e poi applicare l'istogramma delle distanze per trovare centro e raggio di **1** solo cerchio.
Infatti in questo caso l'algoritmo risulta molto più efficace e facilmente "tunabile".

Protocollo di comunicazione

Per trasmettere i dati dalla simulazione all'algoritmo definiamo un protocollo di comunicazione (per adesso molto semplice, si può rendere più raffinato) basato su **header comune:**

- La simulazione salva i dati su file di output binario ("**file.dat**"), scrivendo le coordinate x e y dei punti di hit in formato float, secondo lo schema: x1, y1, x2, y2, ... "file.dat" viene sovrascritto ogni volta che viene eseguita una simulazione.
- Un secondo file di output ("**simul.dat**") contiene le coordinate del centro e del raggio simulato. I dati sono salvati in "append", file non sovrascritto.
- Terzo file di output ("**fit.dat**") contiene i risultati del fit, file non sovrascritto.

Simulazione Monte Carlo (senza rumore)



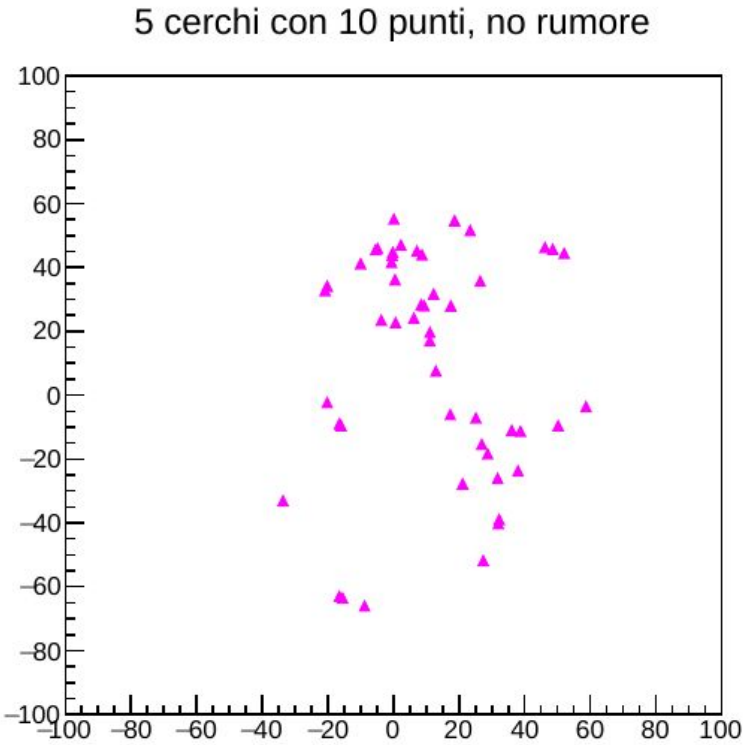
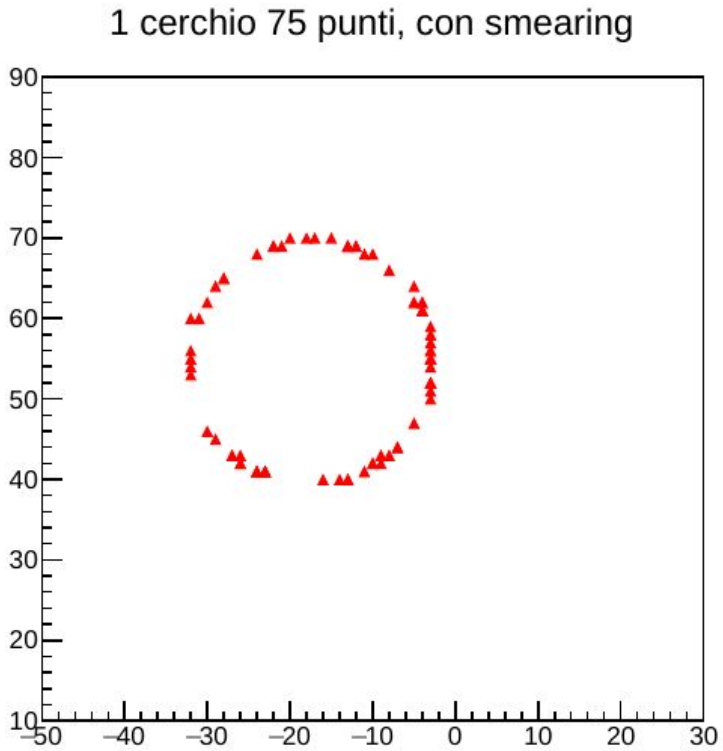


Figura 5

Simulazione Monte Carlo (con rumore)



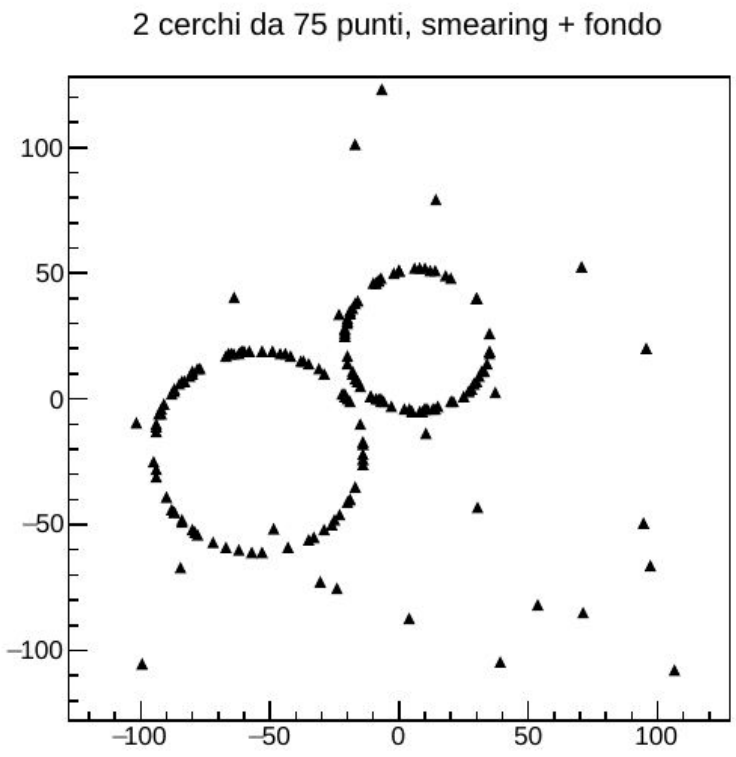


Figura 6

Validazione dell'Algoritmo

Per **validare** l'algoritmo implementato procederemo con i seguenti passi:

1. Test su **1 singolo cerchio** simulato **senza rumore**: da un fit può capitare di trovare più di 1 cerchio, facciamo quindi la media tra i valori trovati dall'algoritmo e la confrontiamo con i valori iniziali della simulazione. Ripetiamo per un **N** volte. Infine **plottiamo** su un **istogramma** questa distanza (valori fittati-iniziali).
2. Test analogo al punto precedente su **1 cerchio** simulato **con rumore.**
3. Medesimo Test su **cerchi multipli** all'interno della stessa simulazione (problema di fare la media tra i valori fittati in modo sensato)

Risultati di Fit su un singolo cerchio (senza rumore)

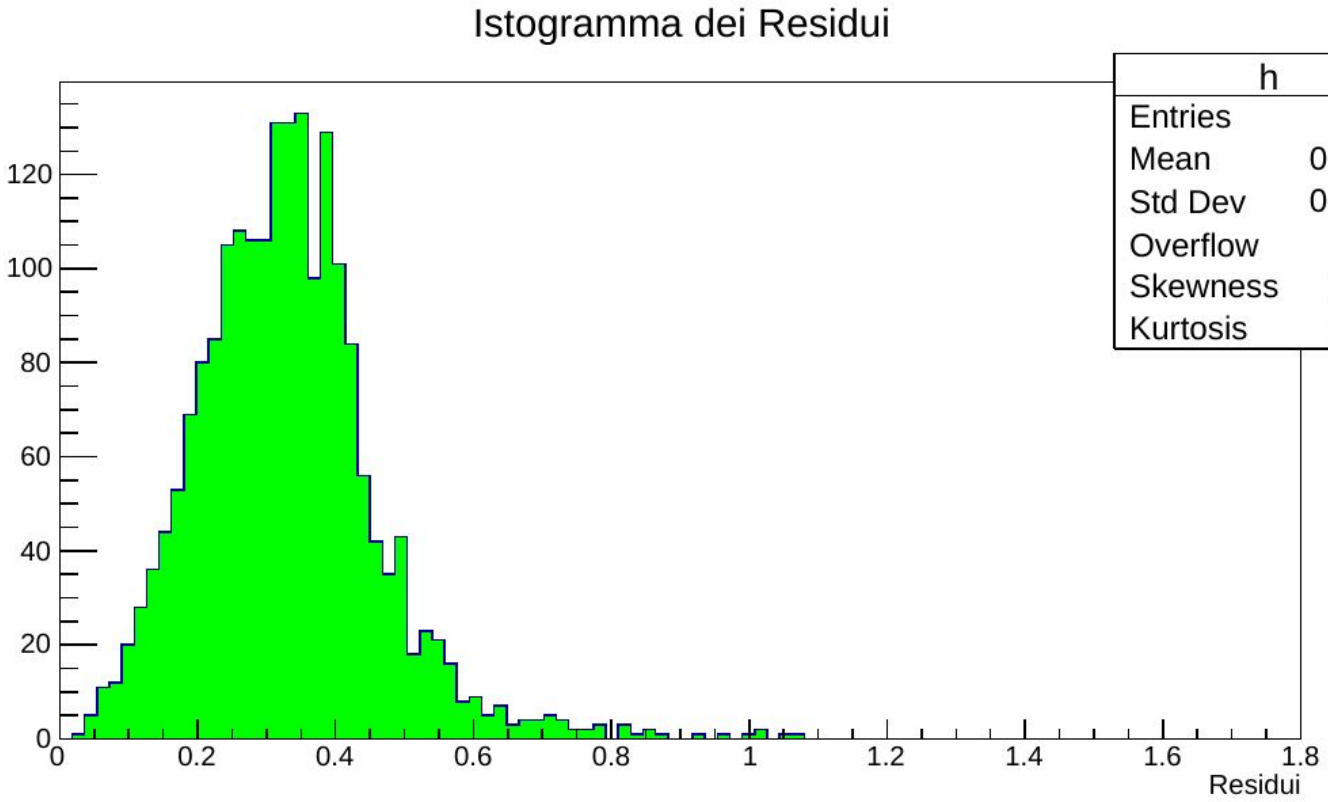
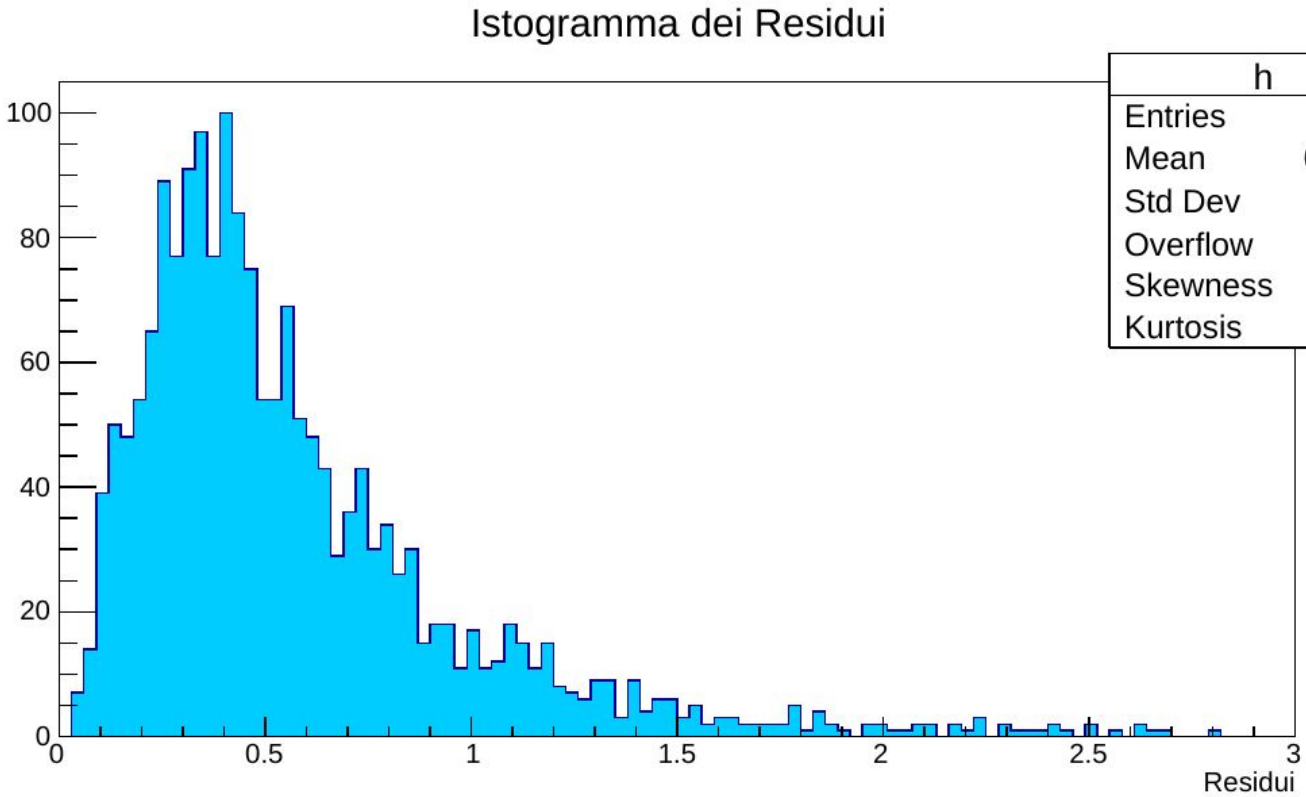


Figura 7
Cerchi: 100 punti
Soglia: 50 punti
Bin size: 1
Fake ratio: 0%

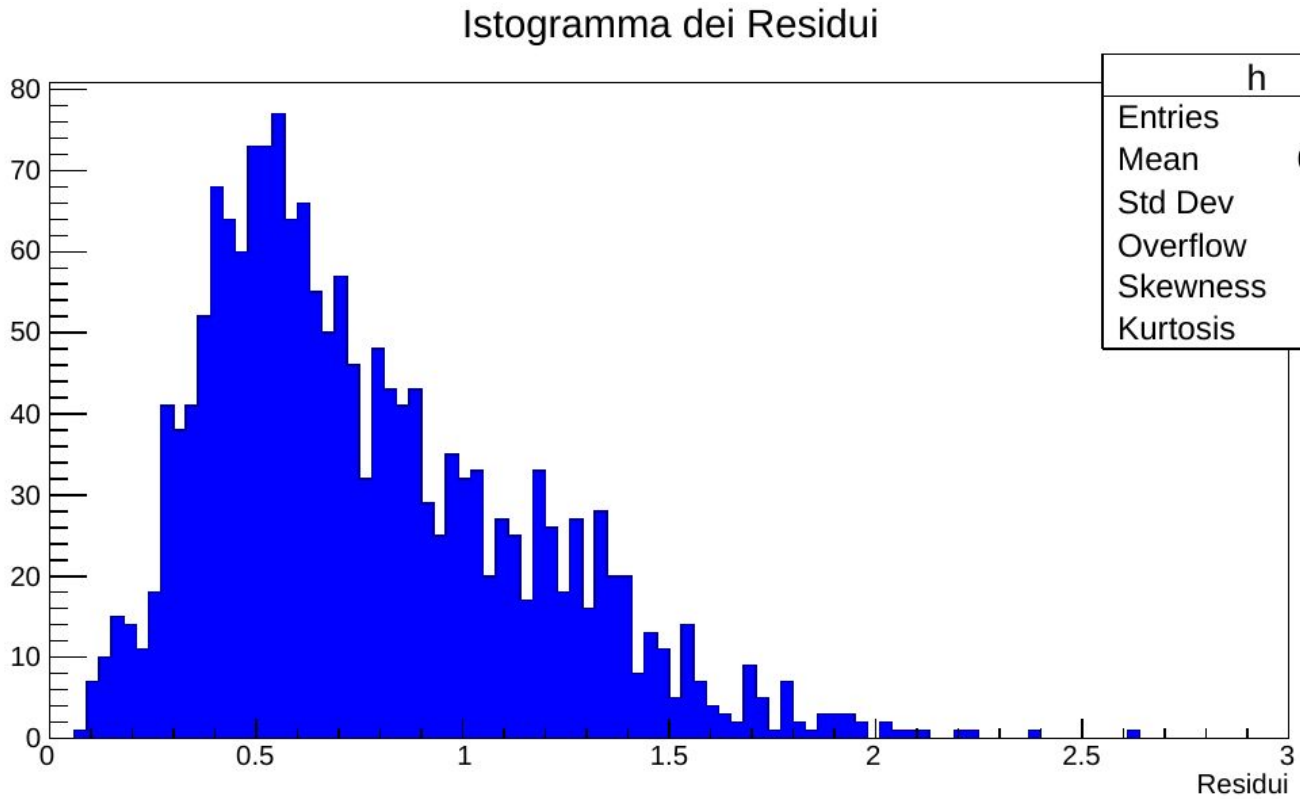
Risultati di Fit su un singolo cerchio (senza rumore) [2]



fake dovuti a cerchi casuali trovati

Figura 8
Cerchi: 10 punti
Soglia: 5 punti
Bin size: 1
Fake ratio: 9.45%

Risultati di Fit su un singolo cerchio (senza rumore) [2]



fake dovuti a cerchi che il fit non è stato in grado di trovare

Figura 9
Cerchi: 10 punti
Soglia: 7 punti
Bin size: 2
Fake ratio: 12.45%

Considerazioni

- Per calcolare i residui ho usato la funzione: sqrt($(x_f - x_s)^2 + (y_f - y_s)^2 + (r_f - r_s)^2$), dove x e y sono le coordinate del centro, r è il raggio, f sta per fittato e s per simulato.
- tutti i fake trovati (nell'istogramma sono i punti di overflow) non sono causati dal fatto che l'algoritmo non ha trovato nessun cerchio, ma sono proprio risultati casuali (ho inserito un printf per controllare). Questo era dovuto al fatto che la simulazione generava valori "particolari" (raggio dei cerchi circa 0 o centri sul bordo del mio spazio). Ora ho corretto la simulazione per evitare che ciò accada e sto aggiornando i grafici.
- **Cerchi** corrisponde al numero di punti per ogni cerchio simulato, **Soglia** è il numero minimo di hit che devono essere presenti nell'istogramma delle distanze per sapere di aver trovato un cerchio, **Bin size** è la dimensione di un canale dell'istogramma delle distanze.

Risultati di Fit su un singolo cerchio (senza rumore) [3]

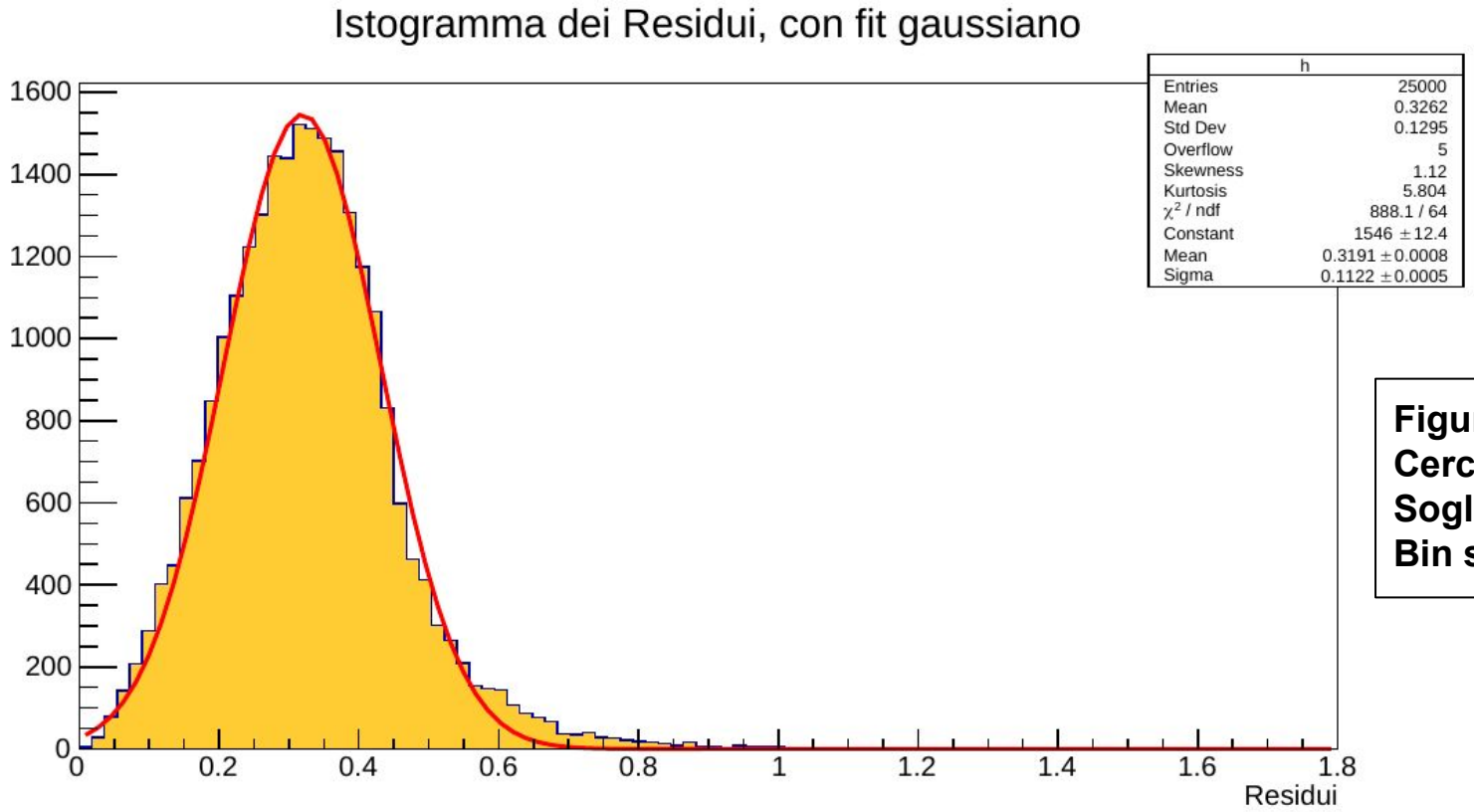


Figura 10
Cerchi: 100 punti
Soglia: 50 punti
Bin size: 1

Risultati di Fit su un singolo cerchio (senza rumore) [4]

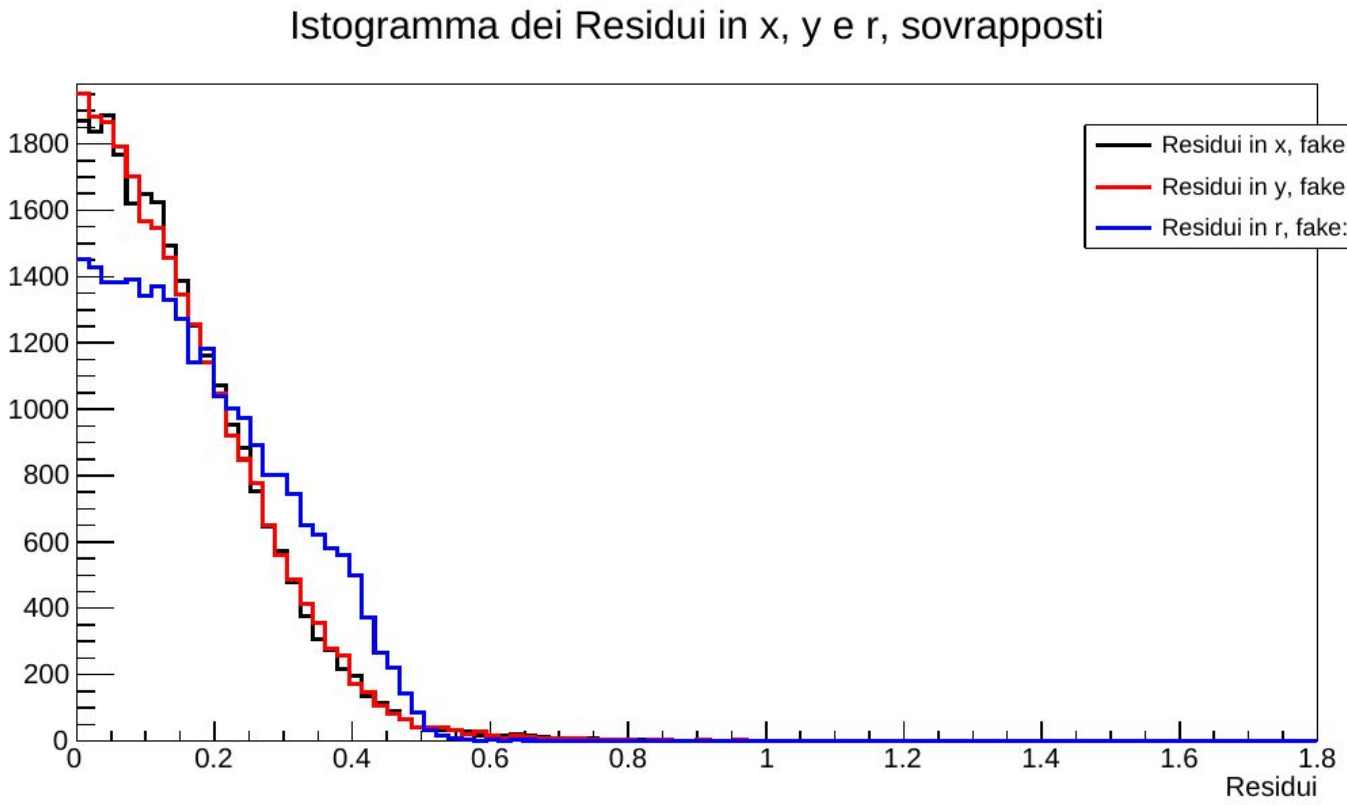


Figura 11
Qua ho usato gli stessi dati della figura 8 ma ho separato i residui:
xcentro : linea **nera**
ycentro : linea **rossa**
raggio : linea **blu**
La figura 9 è la somma in quadratura di questi 3 residui

Risultati di Fit con rumore

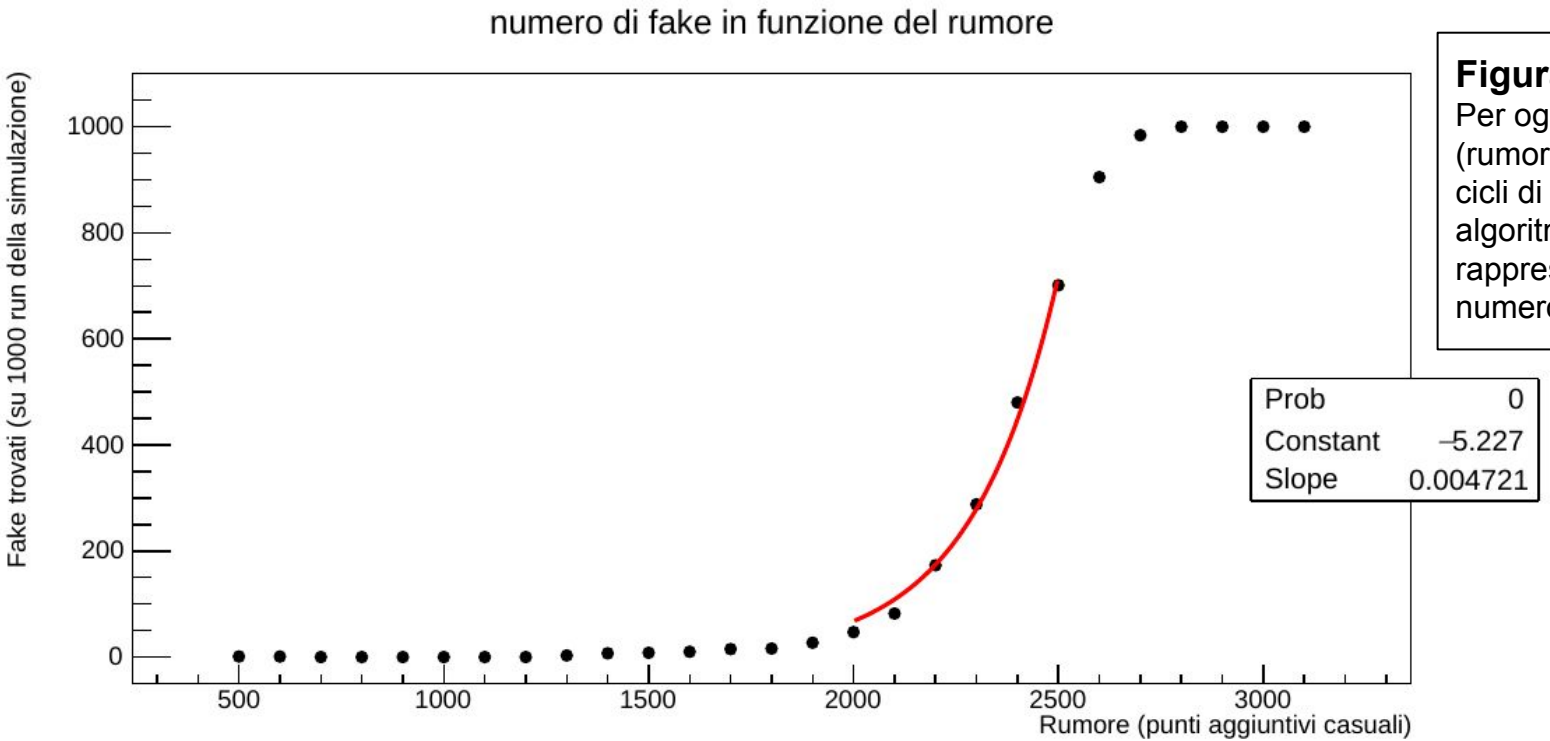


Figura 12
Per ogni valore in x (rumore) ho eseguito 1000 cicli di simulazione + algoritmo di ricostruzione e rappresentato nel grafico il numero di fake trovati.

Fit con più di 1 cerchio simulato

simulazione con 4 cerchi da 100 punti

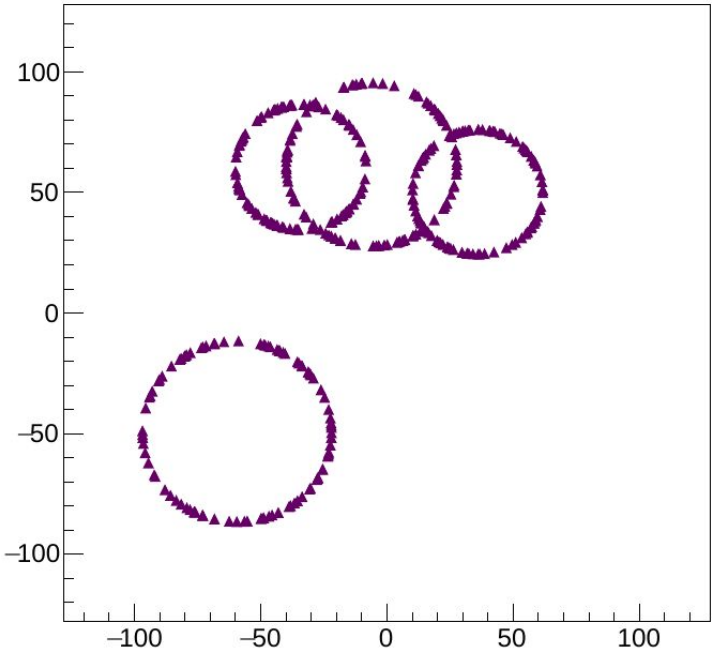


figura13

dati simulati:

x	y	r
35,90911	50,322403	25,812883
-59,393738	-48,944263	37,42229
-6,103016	61,655266	33,80088
-34,15018	60,602737	25,915363

dati fittati:

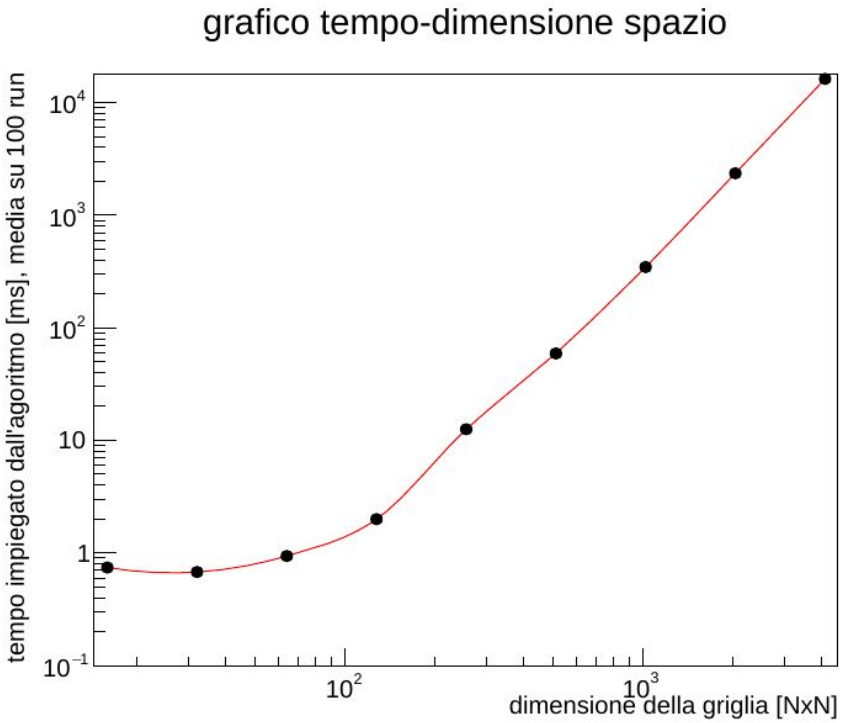
x	y	r
-59	-49	37
-6	61	34
-6	62	34
-6	62	33
36	50	26
36	51	26
-34	61	26

commenti:

L'algoritmo di ricostruzione funziona abbastanza bene fino a 4/5 cerchi nella stessa simulazione.

Non sono stati eseguiti test simili a quelli precedenti perché non sono riuscito a raggruppare i dati fittati nei singoli cerchi, in quanto il numero di cerchi "vicini" tra loro e quindi corrispondente a 1 solo simulato è molto variabile.

Grafici di tempo impiegato dal fit in diverse situazioni



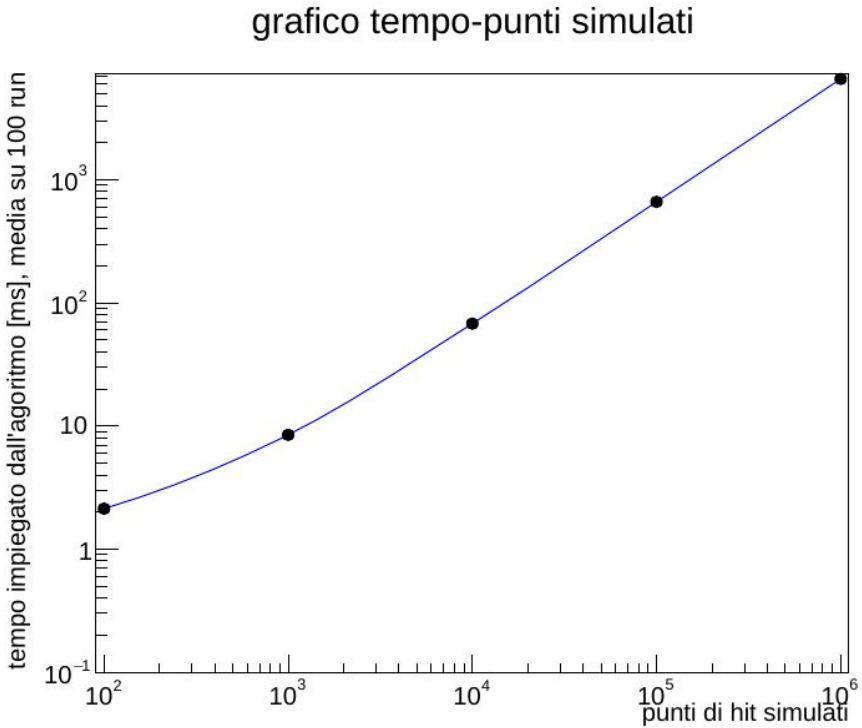


figura 14 a e b

Cosa resta da fare

- Implementare l'algoritmo **"Almagest"** per la divisione dei dati raw in singoli cerchi.
- Migliorare il protocollo di comunicazione, inserendo una **checkword** e **l'ID** dell'evento
- Questo programma è **memory bound**, implementare l'utilizzo della **shared memory** e disaccoppiare il trasferimento dei dati sulla GPU dall'esecuzione del kernel.
- Implementare un metodo automatizzato per la scelta del livello di soglia dell'istogramma delle distanze. Si potrebbe utilizzare come discriminante il fatto che il picco centrale sia 4/5 sigma sopra il fondo.