Alma Mater Studiorum · Università di Bologna

Scuola di Scienze
Dipartimento di Fisica e Astronomia
Corso di Laurea in Fisica

Adattamento e valutazione del software di ricostruzione Patatrack per rivelatori a pixel applicato a dati TrackML

Relatore:
Prof. Francesco Giacomini

Presentata da:
Simone Balducci

Correlatori:
Dott. Felice Pantaleo (CERN)
Dott. Antonio Di Pilato (CASUS)
Dott. Wahid Redjeb (RWTH)

Anno Accademico 2021/2022

Sommario

La ricostruzione delle traiettorie delle particelle prodotte dai vertici di interazione a LHC è di fondamentale importanza per tutti gli esperimenti. Questo passo è uno dei più dispendiosi in termini di tempo e calcolo computazionale nella catena di ricostruzione dell'evento e diventa sempre più complesso con l'aumentare del numero di collisioni. L'esperimento CMS adotta un rivelatore di tracciamento con tecnologia al silicio, dove la parte più interna sfrutta rivelatori con geometria a pixel, mentre la parte esterna utilizza delle strisce di silicio. Per quanto riguarda la ricostruzione nel rivelatore a pixel, sono stati sviluppati diversi algoritmi ottimizzati per fronteggiare l'alto rate di acquisizione dati, sfruttando anche il calcolo parallelo su GPU, con un'efficienza di tracciamento comparabile o superiore agli algoritmi precedentemente utilizzati. Questi nuovi algoritmi sono alla base del software Patatrack per la ricostruzione delle traiettorie.
Il lavoro descritto in questa tesi punta ad adattare Patatrack ad una geometria diversa e più complessa di quella di CMS e di valutarne le prestazioni di fisica e computazionali. Sono stati utilizzati i dati forniti dalla *TrackML challenge*, il cui scopo è incentivare lo sviluppo di nuovi algoritmi di ricostruzione di tracce per gli esperimenti in fisica delle alte energie. È stato condotto uno studio approfondito della nuova geometria per potervi successivamente adattare il software esistente. Infine, la catena di ricostruzione è stata modificata per poter utilizzare i dati forniti dalla TrackML challenge e permettere la ricostruzione delle traiettorie.

Indice

1 LHC e l'esperimento CMS **5**
1.1 Il Large Hadron Collider 5
1.2 L'esperimento CMS 6
1.3 Luminosità, pile up e sezione d'urto 7

2 La ricostruzione delle traiettorie **9**
2.1 Modelli di traiettorie 9
2.1.1 Equazioni del moto 9
2.1.2 Parametrizzazione delle traiettorie 10
2.1.3 Sistema di coordinate usato a CMS 12
2.1.4 Propagazione della traiettoria 12
2.1.5 Traiettorie in campi magnetici omogenei 13
2.1.6 Propagazione degli errori 14
2.2 Qualità delle traiettorie ricostruite 14
2.3 Il software Patatrack 15
2.3.1 Costruzione degli n-tupletti 17

3 Ricostruzione Paratrack per TrackML **21**
3.1 La TrackML challenge 21
3.2 Struttura dei dati 21
3.3 Visualizzazione dei dati 24
3.4 Ricostruzione dei doppietti 26
3.5 Visualizzazione dei doppietti 26
3.6 Ricostruzione delle traiettorie con Patatrack 28

4 Conclusioni e sviluppi futuri **31**

Bibliografia **33**

Ringraziamenti **35**

Capitolo 1

LHC e l'esperimento CMS

1.1 Il Large Hadron Collider

Il *Large Hadron Collider* (LHC) è l'acceleratore di particelle più grande e più potente al mondo [1]. Esso consiste di un anello lungo 27 chilometri, composto da magneti superconduttori con diverse strutture acceleranti, aventi il compito di incrementare l'energia delle particelle durante il loro moto.
Dentro l'acceleratore, i due fasci di protoni ad alta energia (circa $6.5\,TeV$) viaggiano a velocità prossime a quella della luce ($\beta \approx 0.999999991$), prima che vengano fatti collidere. Questi si propagano in direzioni opposte e in condotti diversi, dentro dei tubi che vengono tenuti sotto vuoto. Le particelle di questi fasci vengono guidate dentro l'acceleratore da un campo magnetico di circa 8.3 T che viene mantenuto da elettromagneti superconduttori. Gli elettromagneti sono costruiti da bobine di cavi elettrici in niobio di titanio, che operando in uno stato superconduttivo, minimizzano resistenza e perdita di energia e conducono la corrente elettrica in maniera efficiente. Questo richiede che il magnete venga raffreddato fino a temperature molto basse, attorno a 1.85 K, cioè a temperature più basse di quelle che si trovano nello spazio aperto. Per questo motivo l'acceleratore è collegato in più punti a un sistema di distribuzione di elio liquido, che ha il compito di raffreddare il magnete e altri dispositivi.
Migliaia di magneti di diverse varietà e dimensioni sono usati per direzionare il fascio all'interno dell'acceleratore. Sono presenti 1232 dipoli magnetici di 15 metri in lunghezza che curvano il fascio e altri 392 quadrupoli magnetici, ognuno di 5 o 7 metri in lunghezza, che collimano il fascio. Da qui, i fasci dentro LHC vengono fatti collidere in 4 punti diversi (si veda la Fig. 1.1), corrispondenti alle posizioni dei 4 rivelatori di particelle, ATLAS, CMS, ALICE e LHCb.

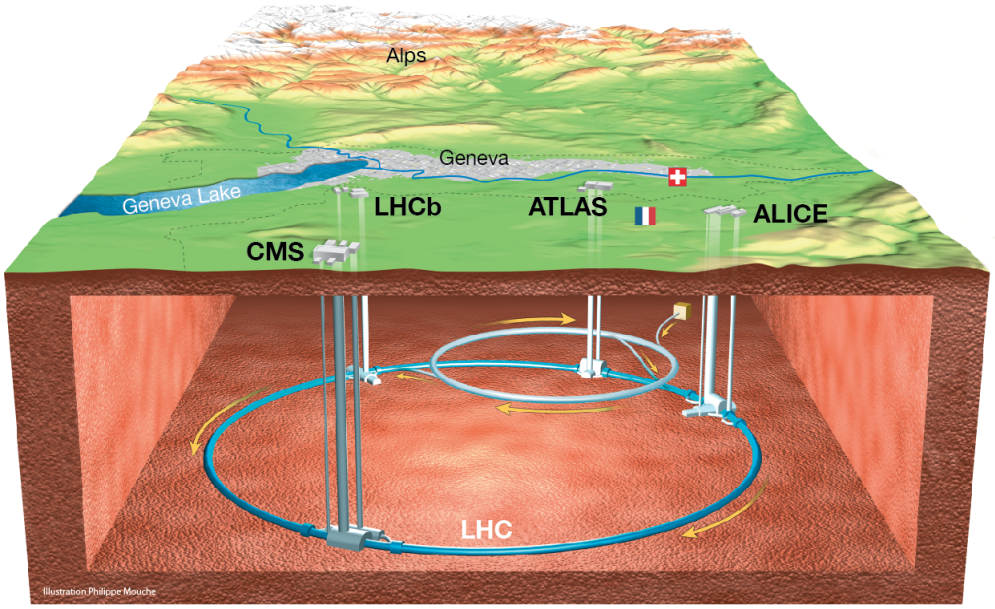

Figura 1.1: Rappresentazione di LHC e dei suoi quattro rivelatori principali.

1.2 L'esperimento CMS

Il *Compact Muon Solenoid* (CMS) è uno dei quattro rivelatori situato in uno dei quattro punti di collisione in LHC. Il programma di fisica di CMS spazia dallo studio del Modello Standard, alla ricerca di nuova fisica oltre il Modello Standard (Beyond Standard Model) e cerca di rispondere ai quesiti ancora aperti, come l'origine della materia oscura, l'asimmetria tra materia e antimateria, e l'inclusione della forza gravitazionale nel Modello Standard [2].
Il rivelatore CMS è costituito da diversi rivelatori posti in maniera concentrica attorno alla linea di fascio. CMS impiega un solenoide superconduttivo generante un campo magnetico di $3.8\,T$ in grado di curvare particelle cariche ad alta energia. Le informazioni provenienti dai diversi rivelatori vengono combinate per fornire una descrizione completa dell'evento. Di seguito una descrizione più dettagliata:

- Curvatura delle particelle
  Dalla curvatura delle particelle si possono ottenere due fondamentali informazioni: identificare la carica delle particelle, in quanto cariche opposte curvano in direzioni opposte. Inoltre, dal momento che la curvatura dipende direttamente dal momento delle particelle, misurarla permette di ottenere una misura del momento.

- Identificazione delle traiettorie
  È necessario identificare le traiettorie percorse dalle particelle, e questo

deve essere fatto con un alto livello di precisione. Questo è fatto mediante un tracciatore al silicio (Tracker), formato da 75 milioni di sensori elettronici disposti in strati concentrici. Quando una particella attraversa i layer (strati) del Tracker, interagisce elettromagneticamente con il silicio e produce un *hit*. Questi hit vengono in seguito collegati per identificare le traiettorie della particella.

- Misura dell'energia
  La conoscenza delle energie delle varie particelle prodotte in ogni collisione è di cruciale importanza per conoscere il processo fisico che ha avuto luogo. Queste informazioni sono raccolte mediante due tipi di calorimetri. Il Calorimetro Elettromagnetico (ECAL) è quello più interno dei due e misura le energie di elettroni e fotoni assorbendoli completamente. Esso è composto da da 76000 cristalli di $PbWO_4$ e si estende fino a $|\eta| < 3$ (Fig 2.4).
  Gli adroni passano attraverso ECAL e interagiscono con il Calorimetro Adronico (HCAL), posto nella regione esterna, adiacente ad ECAL.

- Rivelazione dei muoni
  La particella finale che CMS osserva direttamente è il muone. I muoni hanno una massa di circa 104 MeV (quindi sono 200 volte più pesanti degli elettroni). I muoni non interagiscono con i calorimetri, e per la loro rivelazione sono stati sviluppati dei sotto-rivelatori dedicati.

Il calorimetro ECAL è suddiviso in una sezione "barrel" e due "endcaps". Il barrel cilindrico consiste di 61200 cristalli divisi in 36 "supermoduli", ognuno dei quali pesa tre tonnellate e contiene 1700 cristalli. Gli endcaps, posti perpendicolarmente alla linea di fascio e posizionati alle estremità di CMS sono costituiti da quasi 15000 ulteriori cristalli.

1.3 Luminosità, pile up e sezione d'urto

Negli esperimenti di collisione, le particelle vengono accelerate fino ad ottenere un momento iniziale p_{in} e vengono diffuse in vicinanza dell'origine, con un momento finale $\mathbf{p}^{out}$. Il numero ΔN di particelle diffuse in una certa direzione è proporzionale all'angolo solido $\Delta\Omega$ attorno all'apertura del rivelatore, all'intervallo di tempo considerato Δt e al flusso delle particelle ϕ_{in}. Si ottiene cosi:

$$\Delta N = \frac{d\sigma_{\mathbf{p}^{in}}(\mathbf{n})}{d\Omega}\Delta\Omega\Delta t\Phi^{in} \tag{1.1}$$

dove $d\sigma_{\mathbf{p}^{in}}(\mathbf{n})/d\Omega$ è un fattore di proporzionalità chiamata *sezione d'urto differenziale*, e ha le dimensioni di un'area. Questa grandezza dipende dalla direzione di diffusione e dalla quantità di moto delle particelle incidenti.

Mentre $\Delta\Omega$, Δt e Φ^{in} hanno valori che dipendono dalla configurazione dell'esperimento, la sezione d'urto differenziale ha un significato fisico intrinseco. La sezione d'urto fornisce una misura della probabilità che una particella sia diffusa ad un certo angolo.
Integrando la sezione d'urto differenziale su tutto l'angolo solido si ottiene la sezione d'urto totale:

$$\sigma_{\mathbf{p}^{in}} = \int d^2\mathbf{n} \frac{d\sigma_{\mathbf{p}^{in}}(\mathbf{n})}{d\Omega} \tag{1.2}$$

e questa indica la probabilità che una particella venga diffusa.

In fisica delle particelle la luminosità è il rapporto tra il numero di eventi rivelati in un certo intervallo temporale e la sezione d'urto del processo:

$$L = \frac{1}{\sigma}\frac{dN}{dt} \qquad\qquad [L] = cm^{-2}s^{-1} \tag{1.3}$$

La luminosità è una grandezza utile per quantificare le performance in un acceleratore di particelle, ed è fondamentale per il calcolo delle sezioni d'urto del processo. In particolare tutti gli esperimenti di collisione hanno l'obiettivo di massimizzare la loro luminosità, perché questo comporta un numero maggiore di dati da analizzare, aumentando conseguentemente la probabilità di osservare eventi rari (i.e. eventi previsti da teorie fisiche oltre il Modello Standard).
È importante sottolineare il fatto che la luminosità non indica semplicemente il tasso di collisione, ma indica il numero di particelle che è possibile far passare in una certa superficie in un certo intervallo temporale. Questo è chiaramente legato alla probabilità di collisione perché più particelle vengono inserite in una certa superficie, più aumenta la probabilità che queste collidano tra di loro.
La luminosità raggiunta a LHC è dell'ordine di $7 \times 10^{34}\, cm^{-2}s^{-1}$ [1], e questa produce una media di 600 milioni di collisioni al secondo.

Capitolo 2

La ricostruzione delle traiettorie

2.1 Modelli di traiettorie

2.1.1 Equazioni del moto

Si consideri una particella carica, avente massa m e carica elettrica $Q = qe$, dove q vale ± 1 [3]. Se questa è immersa in un campo magnetostatico $\mathbf{B(r)}$, la sua traiettoria $\mathbf{r}(t)$ è determinata dalle equazioni del moto, ottenibili a partire dalla forza di Lorentz:

$$\mathbf{F} = kq\mathbf{v} \times \mathbf{B}(\mathbf{r}(t)) \tag{2.1}$$

dove k è un fattore di proporzionalità che dipende dalle unita di misura usate. Nel vuoto, si ottiene la seguente forma per la seconda legge di Newton:

$$\frac{d\mathbf{p}}{dt} = kq\mathbf{v}(t) \times \mathbf{B}(\mathbf{r}(t)) \tag{2.2}$$

dove $\mathbf{p}$ è la quantità di moto relativistica, $\mathbf{p} = \gamma m\mathbf{v}$.
Si può inoltre scrivere l'equazione precedente in termini della lunghezza del percorso lungo la traiettoria, $s(t)$, al posto di t, ottenendo cosi l'equazione:

$$\frac{d^2\mathbf{r}}{ds^2} = k\psi\frac{d\mathbf{r}}{ds} \times \mathbf{B}(\mathbf{r}(s)) = \Gamma(s, \mathbf{r}(s), \dot{\mathbf{r}}(s)) \tag{2.3}$$

dove $\psi = q/p$, con $p = |\mathbf{p}|$.
La traiettoria è definita interamente dalle condizioni iniziali, cioè dalle tre coordinate spaziali e le tre componenti della quantità di moto. La collezione $\mathbf{q} = (q_1, q_2, q_3, q_4, q_5, q_6)$ di queste quantità prende il nome di vettore dei parametri iniziali della traiettoria o vettore di stato iniziale.
In presenza di un campo magnetico omogeneo, la traiettoria è un'elica, che si riduce a una linea retta nel limite del campo magnetico che si annulla.
Nel caso di un campo magnetico non omogeneo invece non è possibile trovare una soluzione analitica ed è necessario usare metodi numerici come il metodo di Runge-Kutta.

2.1.2 Parametrizzazione delle traiettorie

In base alla geometria del rivelatore vengono scelte diverse collezioni di parametri della traiettoria. Ciononostante, tutte le traiettorie devono rispettare dei requisiti di base:

- I parametri devono essere continui rispetto a piccoli cambiamenti della traiettoria.
- La scelta dei parametri deve facilitare l'espansione locale dell'equazione della traiettoria in una funzione lineare.
- Le incertezze stocastiche dei parametri della traiettoria devono seguire il più possibile delle distribuzioni gaussiane.

In un detector (rivelatore) *barrel*, cioè un detector in cui i layer sono disposti in una configurazione cilindrica attorno alla linea di fascio, una superficie di riferimento della traiettoria è un cilindro di raggio R, centrato attorno all'asse z, che di solito coincide con la linea di propagazione del fascio. In tal caso i parametri sono definiti dal punto di intersezione tra la traiettoria e il cilindro. Così una possibile scelta di parametrizzazione potrebbe essere la seguente:

$$q_1 = q/p_T, \quad q_2 = \phi, \quad q_3 = \tan\lambda, \quad q_4 = R\Phi, \quad q_5 = z \tag{2.4}$$

dove $p_T = p\cos\lambda$ è il momento trasverso, ϕ è l'angolo azimutale della tangente alla traiettoria nel punto di intersezione P, λ è l'angolo di dip della tangente in P, e infine R, Φ e z sono le coordinate cilindriche di P (Fig. 2.1).

Invece in un sistema di rivelatori basato su rivelatori piani, la più naturale superficie di riferimento è un piano. Tale superficie è determinata unicamente dal versore normale al piano e dalla posizione di un punto di riferimento sul piano.
Un sistema di coordinate locali è definito cosi che l'asse u sia parallelo al vettore normale e che gli assi v e w siano contenuti nel piano. Una scelta naturale di parametri in questo caso può essere:

$$q_1 = \psi, \quad q_2 = \frac{dv}{du}, \quad q_3 = \frac{dw}{du}, \quad q_4 = v, \quad q_5 = w \tag{2.5}$$

dove $\psi = q/p$, dv/du è la tangente dell'angolo compreso tra la proiezione della traiettoria nel piano (u, v) e l'asse u e dw/du è la tangente dell'angolo tra la proiezione della traiettoria nel piano (u, w) e l'asse u.

Un altro sistema di riferimento planare è quello curvilineo, che è utile per trasportare le incertezze dei parametri. Questo sistema di riferimento è sempre ortogonale alla direzione della traiettoria, e la parametrizzazione è:

$$q_1 = \psi, \quad q_2 = \phi, \quad q_3 = \lambda, \quad q_4 = x_\perp, \quad q_5 = y_\perp \tag{2.6}$$

dove $x_\perp$ e $y_\perp$ sono coordinate posizionali ortogonali nel piano.

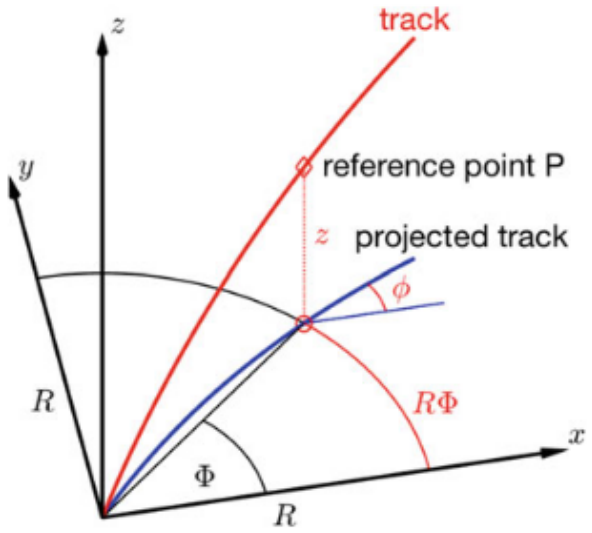

Figura 2.1: Parametrizzazione descritta dall'equazione 2.4. Il parametro $\tan\lambda$ indica l'inclinazione della tangente nel punto di riferimento rispetto al piano (x, y).

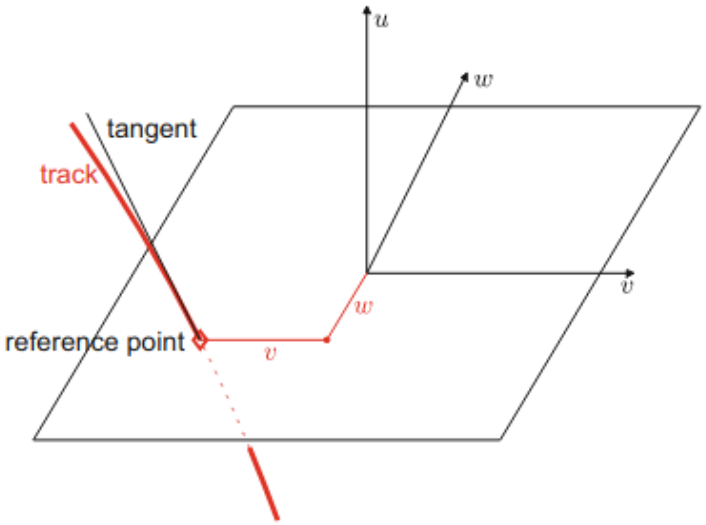

Figura 2.2: Parametrizzazione descritta dall'equazione 2.5. I parametri dv/du e dw/du indicano le direzioni delle tangenti della traiettoria nel punto di riferimento rispetto all'asse u.

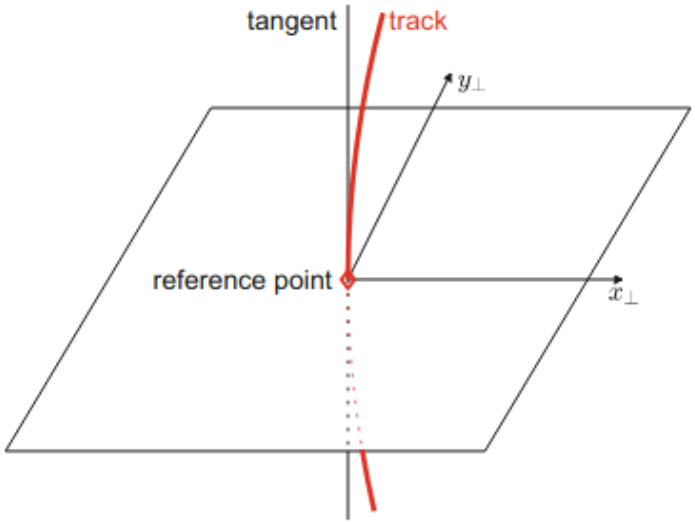

Figura 2.3: Parametrizzazione descritta dall'equazione 2.6. I parametri $x_\perp$ e $y_\perp$ indicano le distanze dal punto di riferimento nel piano perpendicolare alla traiettoria. La direzione della tangente viene misurata in coordinate polari globali.

2.1.3 Sistema di coordinate usato a CMS

Il sistema di coordinate usato nell'esperimento CMS (Fig. 2.4) è un sistema di coordinate cilindrico, orientato in modo che l'asse x punti verso il centro dell'anello LHC, l'asse y punti verso l'alto e l'asse z punti nella direzione del fascio di particelle, diretto verso i monti Jura. Gli eventi in CMS possono anche essere descritti in un sistema di coordinate cilindriche, con variabili $\phi \in (0, 2\pi)$, $\eta \in (0, \infty)$ (il cui segno dipende dalla direzione della particella) e $r \in (0, \infty)$. L'angolo azimutale ϕ è misurato partendo dall'asse x nel piano (x, y), mentre la coordinata radiale, che viene misurata nello stesso piano, viene indicata con R. L'angolo polare θ invece viene misurato nel piano (R, z).
La pseudo-rapidità viene definita come:

$$\eta = -\log\left[\tan\left(\frac{\theta}{2}\right)\right] \tag{2.7}$$

La componente del momento trasverso al fascio di particelle, denotata con p_T, viene calcolata dalle componenti x e y. L'energia trasversa è definita come:

$$E_T = E\sin\theta \tag{2.8}$$

2.1.4 Propagazione della traiettoria

Il propagatore della traiettoria, $\mathbf{f}_{j|i}$, dato dalle soluzioni delle equazioni del moto, descrive la dipendenza funzionale del vettore di stato $\mathbf{q}_j$ su una superficie

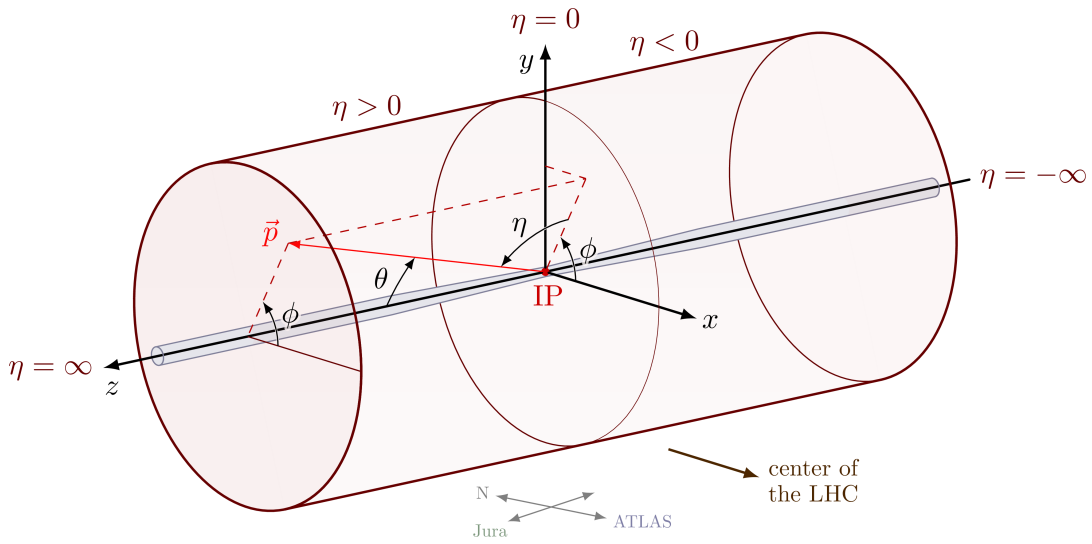

Figura 2.4: Sistema di coordinate cilindrico usato nell'esperimento CMS.

j da un altro vettore di stato, $\mathbf{q}_i$ su un'altra superficie, i.

$$\mathbf{q}_j = \mathbf{f}_{j|i}(\mathbf{q}_i) \tag{2.9}$$

2.1.5 Traiettorie in campi magnetici omogenei

Il propagatore della traiettoria elicoidale prende la soluzione dell'equazione 2.3. La soluzione si può scrivere come:

$$\mathbf{r}(s) = \mathbf{r}_0 + \frac{\delta}{K}(\theta - \sin\theta) \cdot \mathbf{h} + \frac{\sin\theta}{K} \cdot \mathbf{t}_0 + \frac{\alpha}{K}(1 - \cos\theta) \cdot \mathbf{n}_0 \tag{2.10}$$

dove $\mathbf{r}(s)$ è il vettore posizione sull'elica, s è la lunghezza percorsa rispetto al punto iniziale $\mathbf{r}_0$, $\mathbf{h} = \mathbf{B}/B$ è il campo magnetico normalizzato, $\mathbf{t}$ è il vettore tangente alla traiettoria, $\delta = \mathbf{h} \cdot \mathbf{t}$ indica la componente del campo magnetico lungo la tangente alla traiettoria, $\mathbf{n} = (\mathbf{h} \times \mathbf{t})/|\mathbf{h} \times \mathbf{t}|$ è il versore normale alla traiettoria e alla direzione del campo magnetico, e infine $K = -k\psi|\mathbf{B}|$, dove k e ψ sono i parametri definiti nel paragrafo (2.1.1). Di seguito il pedice 0 indicherà sempre che la quantità in questione si riferisce a $s = 0$.
L'equazione per il versore $\mathbf{t}$ in funzione di s è:

$$\mathbf{t}(s) = \frac{d\mathbf{r}(s)}{ds} = \delta(1 - \cos\theta) \cdot \mathbf{h} + \cos\theta \cdot \mathbf{t}_0 + \alpha \sin\theta \cdot \mathbf{n}_0 \tag{2.11}$$

Combinando l'equazione 2.10 e 2.11 si possono ottenere i valori di tutti i parametri della traiettoria per ogni valore di s.
Nel modello elicoidale, il momento è costante per ogni s.

2.1.6 Propagazione degli errori

Per un algoritmo di ricostruzione è fondamentale il trasporto della matrice di covarianza dei parametri della traiettoria. Con trasporto si intende passare da una superficie a un'altra mantenendo gli stessi parametri.

Questo processo è semplice quando il vettore dei parametri per la traiettoria trasformata è una funzione strettamente lineare dei parametri iniziali. Tuttavia, in genere, tale trasformazione non è lineare, quindi il processo di propagazione degli errori diventa più complicato. La soluzione più comune consiste nel propagare gli errori linearmente.
Data la matrice Jacobiana di $\mathbf{f}_{j|i}$:

$$\mathbf{F}_{j|i} = \frac{\partial \mathbf{q}_j}{\partial \mathbf{q}_i} \tag{2.12}$$

la matrice di covarianza $\mathbf{C}_i$ dei parametri sulla superficie i è trasportata sulla superficie j secondo:

$$\mathbf{C}_j \approx \mathbf{F}_{j|i}\mathbf{C}_i\mathbf{F}_{j|i}^T \tag{2.13}$$

2.2 Qualità delle traiettorie ricostruite

Mediante degli algoritmi di clustering [4] avviene quella che si chiama *ricostruzione locale*, ovvero la ricostruzione degli hit. In seguito si ha la ricostruzione delle traiettorie associate agli hit ricostruiti.
La qualità delle traiettorie ricostruite si ottiene mediante un confronto con le traiettorie simulate [5]. Queste sono le traiettorie per cui è nota esattamente l'associazione tra gli hit e le particelle.

I criteri principali per determinare la qualità della traiettoria ricostruita, che può essere usato per quantificare le performance degli algoritmi di ricostruzione sono:

- Efficienza
- Fake-rate
- Tempo di esecuzione

Andando per ordine: L'efficienza indica la frazione di traiettorie simulate, N_{sim}, che sono state associate almeno a una traiettoria ricostruita, N_{rec}.

$$eff = \frac{N_{rec}}{N_{sim}} \tag{2.14}$$

Una traiettoria ricostruita è associata a una simulata se più del 75% degli hit che contiene appartengono alla stessa traiettoria simulata.

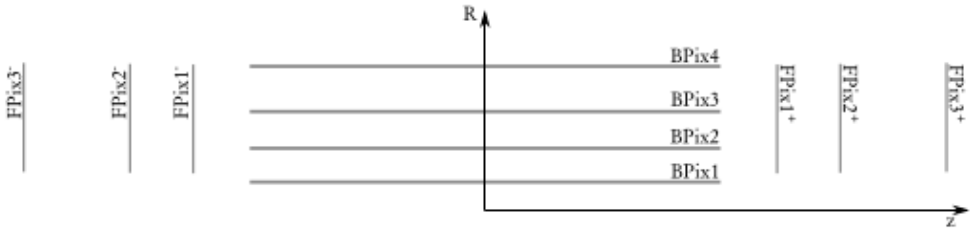

Figura 2.5: Schema dei layer del rivelatore di CMS. I layer indicati con *BPix** sono i barrel, mentre quelli indicati con *FPix** sono i dischi.

Il fake-rate è definito come la frazione delle traiettorie ricostruite che non sono associate unicamente a una traiettoria simulata. Nel caso di una traiettoria falsa, l'insieme di hit usato per ricostruire la traiettoria non appartiene alla stessa traiettoria simulata.
Il tempo di esecuzione è il tempo che il processore ha speso ricostruendo le traiettorie partendo dagli hit.

L'efficienza e il fake-rate vengono spesso chiamate *performance fisiche*, mentre il tempo di esecuzione viene chiamato *performance computazionale.*

2.3 Il software Patatrack

Il processo di ricostruzione delle traiettorie diventa più complicato all'aumentare del numero di vertici e di traiettorie, rendendo il riconoscimento dei pattern e la classificazione degli hit prodotti dalla stessa particella carica un problema combinatoriale più complesso [4].
Per affrontare questo problema è stato sviluppato Patatrack, un algoritmo che esegue la ricostruzione delle traiettorie in parallelo su GPU (Graphics Processing Units), hardware ottimizzato per calcolo parallelo, a partire dai dati raw (cioè i dati in uscita dal sistema di acquisizione che ancora non sono stati processati). Inizialmente, i segnali analogici generati dalle particelle cariche che attraversano i detector sono digitalizzati dall'elettronica in uscita e raggruppati per minimizzare il tasso di dati in uscita. Il primo passo per la ricostruzione delle traiettorie consiste nella ricostruzione locale, cioè la ricostruzione delle informazioni riguardanti gli hit nel detector.
In questa fase, le informazioni digitalizzate vengono analizzate e interpretate per creare i *digi*. Ogni *digi* rappresenta un singolo pixel con una carica superiore al livello di soglia del rumore, e contiene informazioni sulla sua posizione. Questo processo è parallelizzato su due livelli: l'informazione che arriva da diversi moduli è elaborata in parallelo da blocchi di threads diversi, mentre ogni digi in ciascun modulo viene elaborato da un thread diverso.
I digi in ogni modulo sono disposti su una griglia bidimensionale usando le informazioni di riga, colonna e indice. A ogni digi viene poi assegnato un thread,

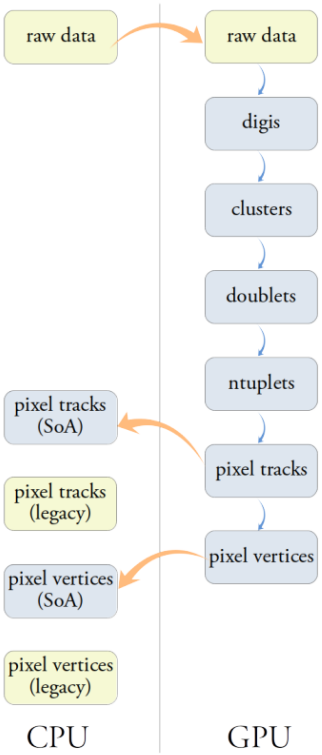

Figura 2.6: Schema che mostra i passi seguiti dal software Patatrack per la ricostruzione delle traiettorie. Si noti come all'inizio del processo si passi dai dati in formato adatto alla CPU a un formato adatto alla GPU.

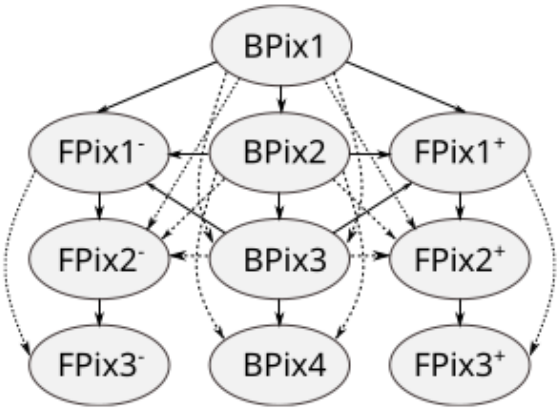

Figura 2.7: Schema che mostra come vengono creati i doublets. I doublets vengono creati automaticamente per layer che sono adiacenti, e queste coppie sono indicate da frecce continue, ma per tenere conto dell'inefficienza dei rivelatori, vengono considerate anche coppie di layer non adiacenti, che in figura sono indicate da frecce tratteggiate

e se due o più digi adiacenti vengono assegnati allo stesso cluster, quello con l'indice minore diventa il seme (*seed*) dell'altro. Questa procedura è ripetuta fino a che tutti i digi sono stati assegnati a un seme, e mediante questo processo iterativo, digi vicini vengono raggruppati per formare cluster. Infine, la forma dei cluster risultanti e le cariche dei digi sono usate per determinare le posizioni di impatto e le loro incertezze sulle coordinate locali del modulo.

2.3.1 Costruzione degli n-tupletti

I clusters creati nella prima fase di ricostruzione locale sono collegati tra di loro per formare degli n-tupletti, che sono in seguito fittati per ottenere i parametri finali della traiettoria. La costruzione degli n-tupletti segue le seguenti fasi:

- Creazione dei doppietti (*doublets*)
- Connessione dei doppietti
- Identificazione dei *root doublets*
- *Depth-first-search* (DFS) per ogni root doublet

I doublets sono creati connettendo hit appartenenti a layer adiacenti del detector. Tuttavia, per tenere conto dell'inefficienza dei layer del detector vengono considerati anche doublets tra layer non adiacenti (Fig. 2.7). In questo modo il numero di combinazioni di hit aumenta drasticamente; pertanto è necessario applicare alcuni criteri di selezione:

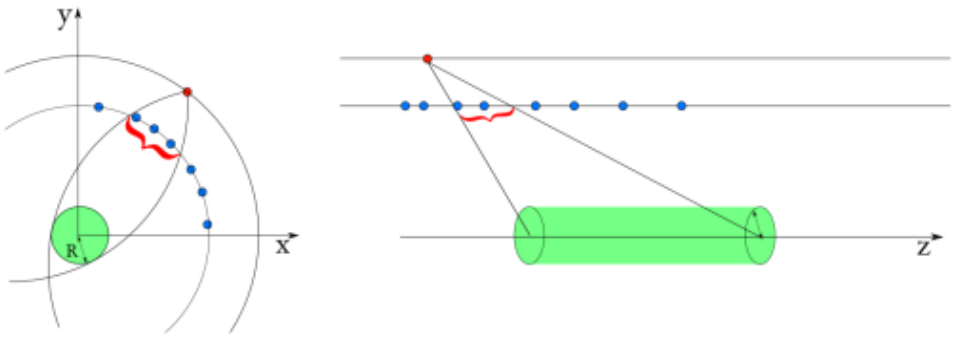

Figura 2.8: Rappresentazione dei criteri di selezione delle traiettorie mediante il raggio massimo, R_{max}, e la distanza massima, z_{max}. Nella figura a sinistra le particelle devono essere contenute in una certa sezione attorno alla linea del fascio, descritta dalla regione verde, e questa è determinata dal raggio massimo. Nella figura a destra le particelle devono essere contenute in un certo intervallo spaziale lungo la linea del fascio determinato.

- P_T^{min}: Cercare traiettorie con momento trasverso basso può avere un costo computazionale molto elevato. Porre un valore minimo di soglia riduce la possibile curvatura, riducendo così il numero di possibili combinazioni di hit.

- R^{max} e z^{max}: Queste grandezze rappresentano la massima distanza trasversa e la distanza longitudinale di massima vicinanza rispetto alla linea del fascio. Cercare traiettorie con un valore di R^{max} maggiore di 1 mm porta a un aumento delle combinazioni, quindi è conveniente applicare questa condizione.

- n_{hit}: Richiedere un numero alto di hit negli n-tupletti porta a produrre traiettorie più pure, mentre un numero di hit minore produce efficienze più alte, al prezzo di aumentare il tasso di traiettorie sbagliate.

Gli hit in ogni layer vengono ordinati secondo l'angolo azimutale ϕ, in modo da semplificare la ricerca di hit associati alla stessa particella e migliorando così le performance computazionali. Tale scelta è motivata dal fatto che, perché due hit siano connessi (e quindi appartenenti alla stessa traiettoria), devono avere un valore di ϕ simile.
La ricerca di coppie di hit compatibili è eseguita in parallelo da threads diversi, ognuno dei quali parte da hit esterni diversi. Una coppia di hit interni ed esterni (ogni doppietto è formato da un hit più vicino alla linea del fascio e da uno più lontano) che rispettano i criteri di allineamento e hanno dimensioni di cluster compatibili lungo la direzione z formano un doublet.
I doublets che hanno un hit in comune vengono testati per verificare se ci sia compatibilità per formare un tripletto. La compatibilità richiede che i tre hit siano allineati nel piano $R - Z$, e che la circonferenza passante attraverso esse

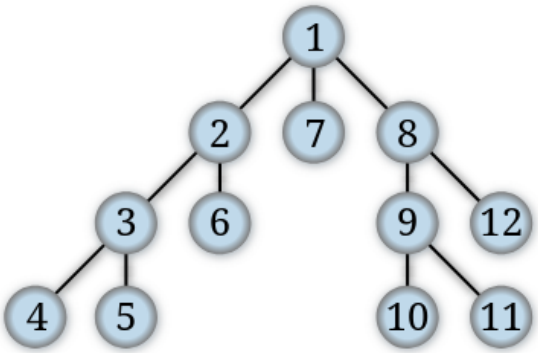

Figura 2.9: Ordine di visita dei nodi di un grafo per un algoritmo DFS, partendo dal nodo 1.

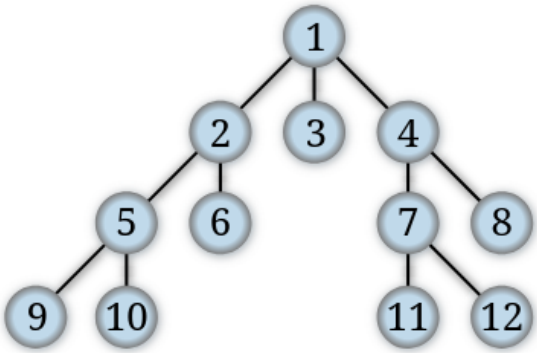

Figura 2.10: Ordine di visita dei nodi di un grafo per un algoritmo BFS, partendo dal nodo 1.

intersechi la zona di compatibilità definita da R_{max}. Tutti i doublets da tutte le coppie di layer vengono testati in parallelo.
Tutti i doublets che hanno l'hit più interno sul layer *BPix1* (Fig. 2.5) sono marcati come *root doublets*. Per ricostruire i tripletti più esterni, anche i doublets che partono dal layer *BPix2* o dai due layer *FPix1* e senza vicini interni sono marcati come *root*. Ogni root doublets in seguito è assegnato a un thread diverso che esegue un DFS.
Un DFS è un algoritmo di ricerca su grafo. La caratteristica principale è quella di visitare nodi che si trovano sullo stesso ramo, e quindi l'algoritmo può visitare nodi lontani dall'origine prima di averne visitati altri più vicini ad essa (Fig. 2.9). L'alternativa a un DFS è un BFS (*breadth-first search*), ovvero una ricerca in ampiezza, che consiste nel visitare tutti i nodi contenuti in un certo raggio e solo successivamente estendere questo raggio e visitare i nodi più lontani (Fig. 2.10). Si preferisce un DFS per la maggiore velocità dell'algoritmo.

Si esegue un DFS perché è preferibile cercare tutti gli n-tupletti fino a raggiungere tutti gli N hit. Il vantaggio di questo metodo è che i gruppi di tripletti

e quadrupletti sono disgiunti, dal momento che i tripletti sono gruppi di hit che non possono essere completati per diventare quadrupletti.

Capitolo 3

Ricostruzione Paratrack per TrackML

3.1 La TrackML challenge

Per affrontare l'aumento della complessità computazionale richiesta dagli esperimenti CMS e ATLAS si è pensato di collaborare con la comunità internazionale che si occupa di sviluppo software. Questo obiettivo è stato raggiunto mediante la Tracking Machine Learning challenge (TrackML) [6], che è stata pubblicata sulla piattaforma Kaggle [7] (una piattaforma online che si occupa di programmazione, data science e machine learning). In questa challenge vegnono forniti dei dati generati da ACTS [8], un simulatore di tracking molto accurato, che si basa su un tipico rivelatore al silicio di LHC, formato da 9 volumi, di cui 6 dischi e 3 barrel (Fig. 3.1). Ogni volume è contiene un numero di layer compreso tra 2 e 7.

L'obiettivo della challenge è la ricostruzione delle traiettorie delle particelle (Fig. 3.2).

Le soluzioni proposte vengono valutate utilizzando due metriche: l'accuratezza e il throughput.

Per l'accuratezza viene assegnato un punteggio che rappresenta la frazione di hit che sono stati assegnati alla traiettoria corretta, con un meccanismo di peso che favorisce le traiettorie ad alta quantità di moto e gli hit nei layer più interni e più esterni.

Per il throughput il punteggio assegnato si basa principalmente sui tempi di esecuzione degli algoritmi di ricostruzione su CPU.

3.2 Struttura dei dati

Nella competizione sono forniti i dati relativi a 9000 eventi. Questi dati contengono le informazioni di "verità", cioè le informazioni che si ottengono ricostruendo le traiettorie, come la particella che ha generato un certo hit e la sua

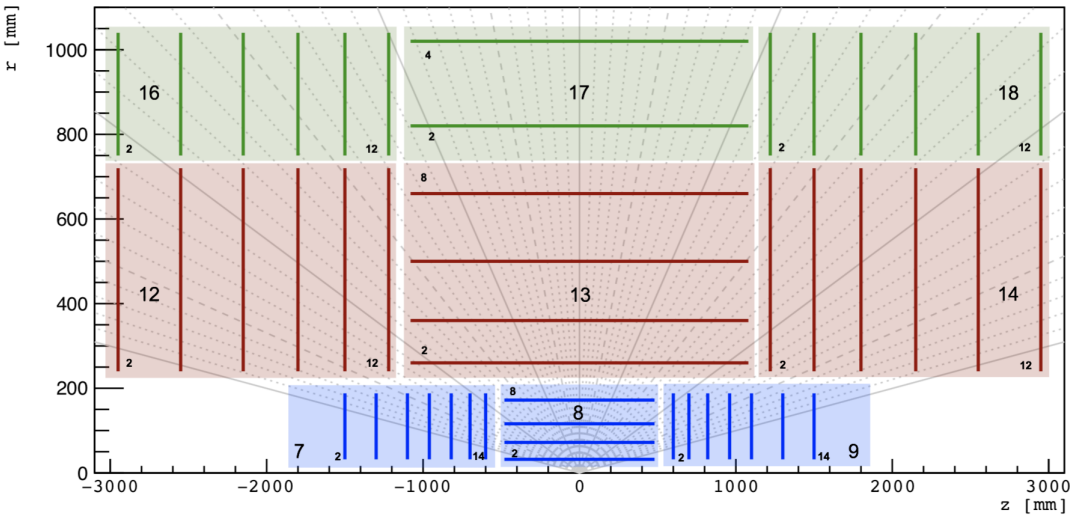

Figura 3.1: Figura che mostra la struttura radiale del detector su cui sono stati generati i dati forniti nella TrackML challenge.

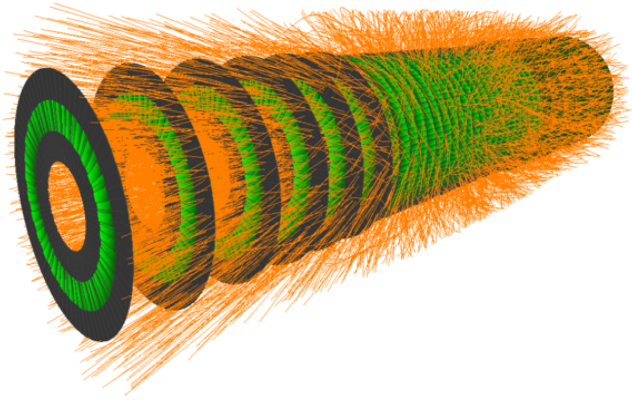

Figura 3.2: Rappresentazione del detector su cui si basa la TrackML challenge e delle traiettorie ricostruite.

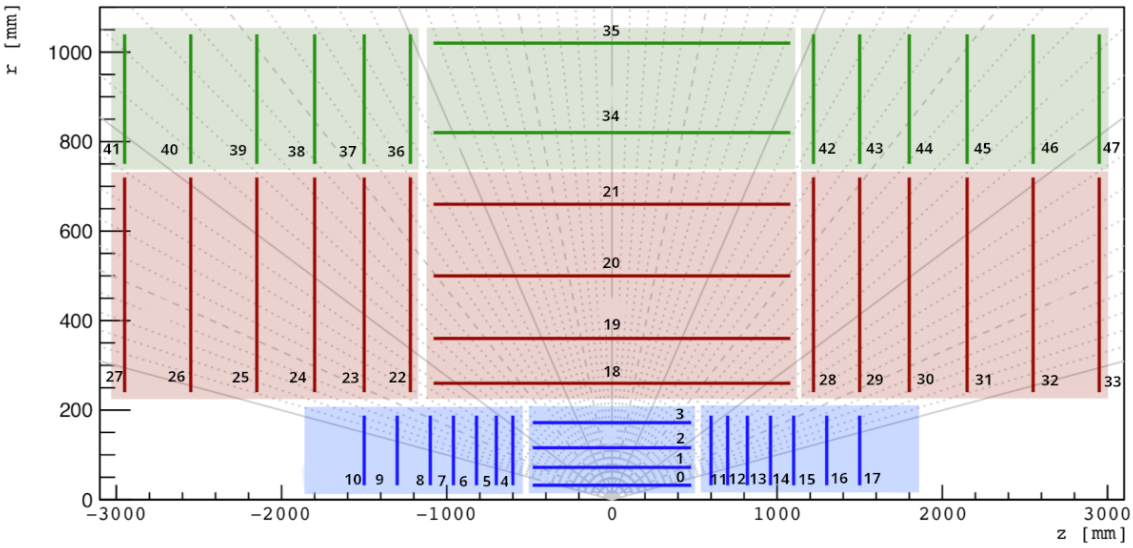

Figura 3.3: Mappatura uasta per assegnare l'indice globale a ogni layer del detector.

vera quantità di moto.
I dati contengono le seguenti informazioni:

- hit, che contengono tutte le informazioni sugli hit registrati dal rivelatore, in particolare le coordinate nel sistema di riferimento cartesiano e il *volume-id* e il *layer-id*, che sono gli identificativi numerici per il volume e il layer in cui l'hit è stato registrato..

- particle, che contengono le informazioni su tutti i tipi di particelle che sono stati identificati nel singolo evento, come la carica elettrica e la quantità di moto. A ogni particella viene assegnato un identificativo intero che prende il nome di *particle-id*.

- truth, che contengono le informazioni di "verità" citate sopra.

Questi file sono stati letti utilizzando la libreria *pandas* [9] di Python, e in seguito usati per creare dei dataframe.
Per rendere la gestione dei dati più semplice e per adattare gli stessi al software Patatrack, ad ogni hit è stato assegnato un indice globale, chiamato *globalIndex*.
Questa nuova indicizzazione è stata implementata mediante una mappa che individua in maniera univoca un indice globale per ogni layer di ogni volume 3.3.

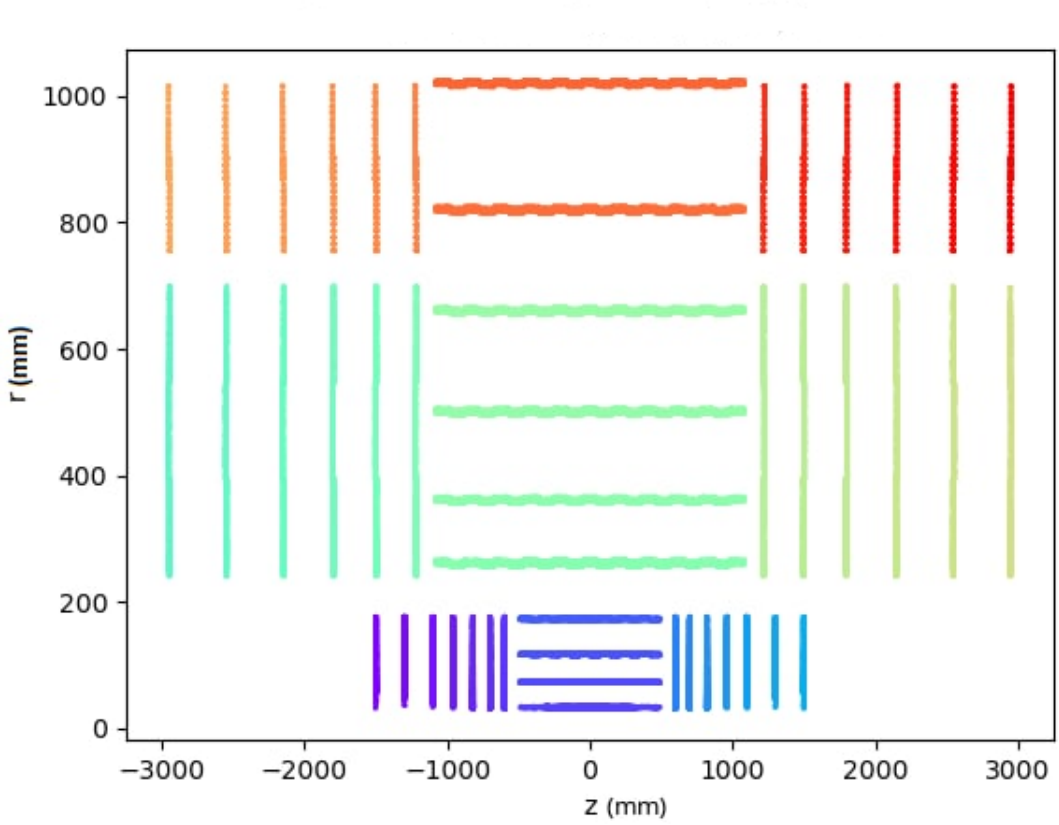

Figura 3.4: Visualizzazione della struttura del detector mediante gli hit nei dati forniti.

3.3 Visualizzazione dei dati

Analizzando tutti gli *hit* è possibile organizzarli in base al layer del detector in cui sono stati registrati. Visualizzando gli hit di tutte le particelle di un evento, come in Fig. 3.4, è possibile ricostruire approssimativamente (in base alla quantità di dati disponibili) la forma dell'intero detector, che deve chiaramente essere identica a quella in Fig. 3.1. Sfruttando la truth si possono associare a ogni particella tutti gli hit che corrispondono alla stessa, potendo così visualizzare la sua traiettoria (Fig. 3.5). È inoltre possibile osservare la forma di un singolo layer visualizzando tutti gli hit su esso disposti/registrati, come nella Fig. 3.6. Infine, dal momento che le fasi successive si basano principalmente sul C++, è necessario trovare un modo per interfacciare i dati letti e analizzati in Python con C++. Il metodo scelto consiste nello scrivere tutti i dati necessari su dei file *.dat*, in maniera tale che questi possano essere letti successivamente in C++.

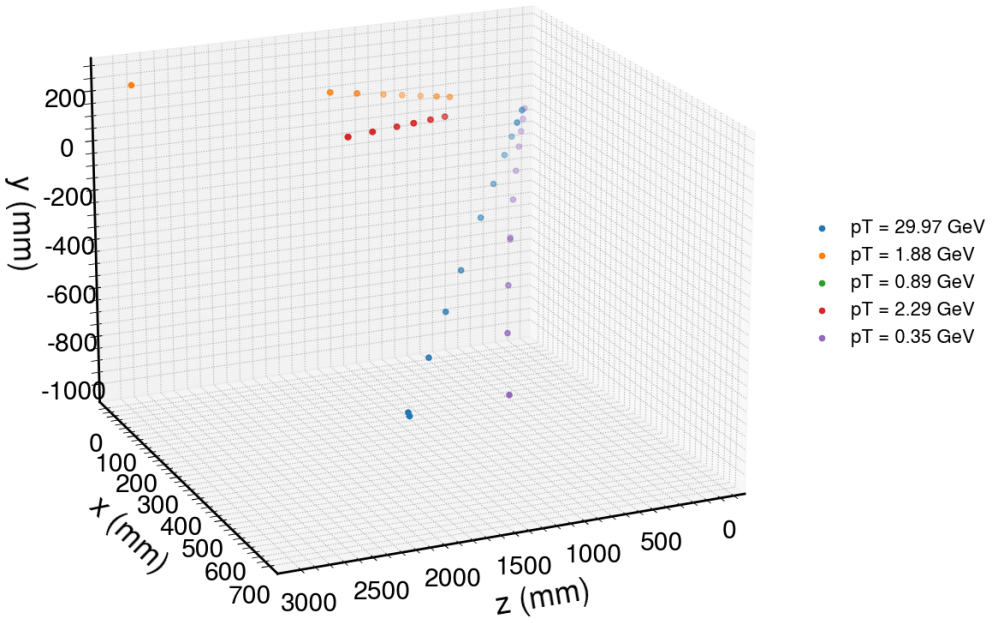

Figura 3.5: Visualizzazione delle traiettorie legate alle prime cinque particelle. Nella legenda, pT indica il momento trasverso.

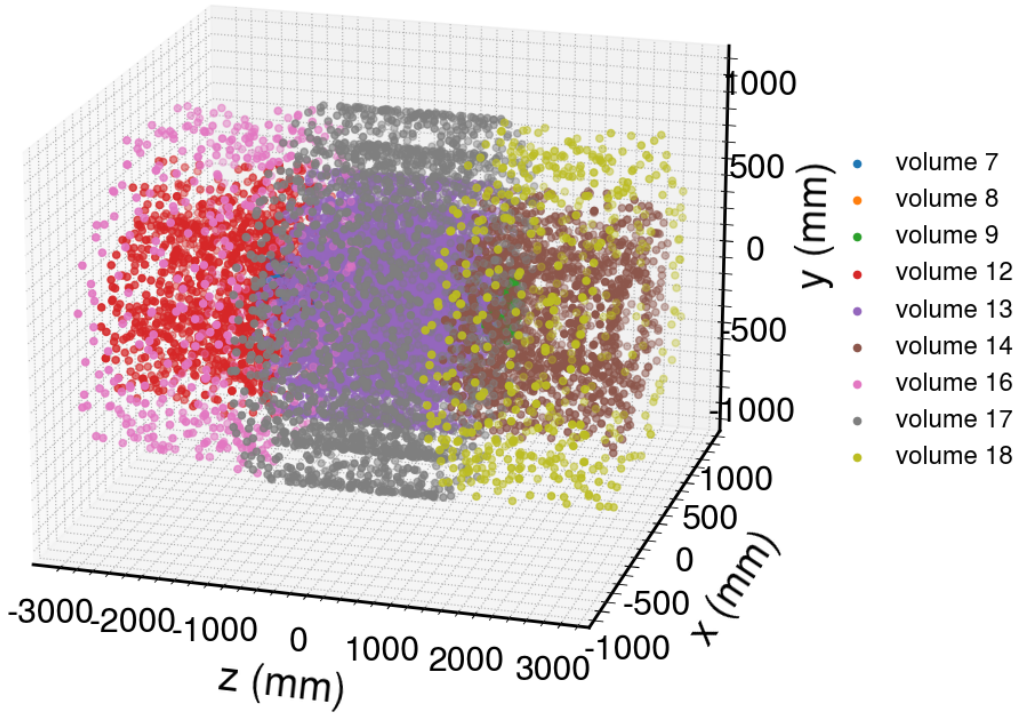

Figura 3.6: Visualizzazione hit registrati su tutto il detector.

3.4 Ricostruzione dei doppietti

Per ricostruire le traiettorie di ogni particella è innanzitutto necessario creare per ognuna di esse i *doublets*, cioè dei doppietti di hit consecutivi.
Per ridurre il numero di doppietti possibili (e il carico computazionale che altrimenti comporterebbe) si può sfruttare l'informazione *truth* per individuare quali coppie di layer registrano un numero elevato di doppietti e che, quindi, si vogliono considerare per la fase successiva della ricostruzione.
Per comodità si definisce un indice ottenuto considerando tutte le combinazioni tra layer.
In seguito si passa a costruire i doppietti per ogni particella.
Considerando i valori univoci del *particle-id* è possibile esaminare tutti gli hit e dividerli in base alle particelle a cui sono associati.

Analizzando le occorrenze dei doppietti ottenuti per tutte le particelle e su tutti gli eventi si trovano degli andamenti che rispecchiano quelli attesi.

- Le colonne più popolate sono quelle corrispondenti agli hit in layer dello stesso volume. In più si nota che si hanno molti hit nei primi due volumi che sono i più vicini al punto di interazione e alla linea del fascio, e che le occorrenze diminuiscono man mano che ci si allontana da tale sorgente (Fig. 3.7).

- A pari distanza dalla linea del fascio, i barrel hanno più doppietti, dal momento che sono quelli che si trovano immediatamente davanti alla sorgente.

3.5 Visualizzazione dei doppietti

Per capire quali doppietti sia necessario considerare e quali possano essere trascurati si utilizza il grafico in figura 3.7.
Si osserva che, come atteso, i volumi molto lontani tra loro non hanno doublets associati. Guardando infatti il grafico in Fig. 3.1 si nota che, ad esempio, per avere un doppietto tra il volume 7 e il volume 16, una particella carica dovrebbe attraversare diversi layer del rivelatore senza generare alcun segnale. Tuttavia, la probabilità che questo accada è estremamente bassa, data l'alta efficienza dei layer del rivelatore.
Dopo aver scartato le coppie di volumi per le quali il numero di doppietti è trascurabile, si passa ad analizzare le occorrenze delle coppie di layer per ogni coppia di volumi.
Prendendo ad esempio il volume 7 (Fig. 3.8), si nota che la maggior parte dei doublets sono formati da hit su layer adiacenti. Si osserva inoltre un numero di occorrenze molto alto per il doppietto $(4-0)$, che corrisponde al primo disk del volume 7 e al primo barrel del volume 8. Doppietti sulle restanti coppie di

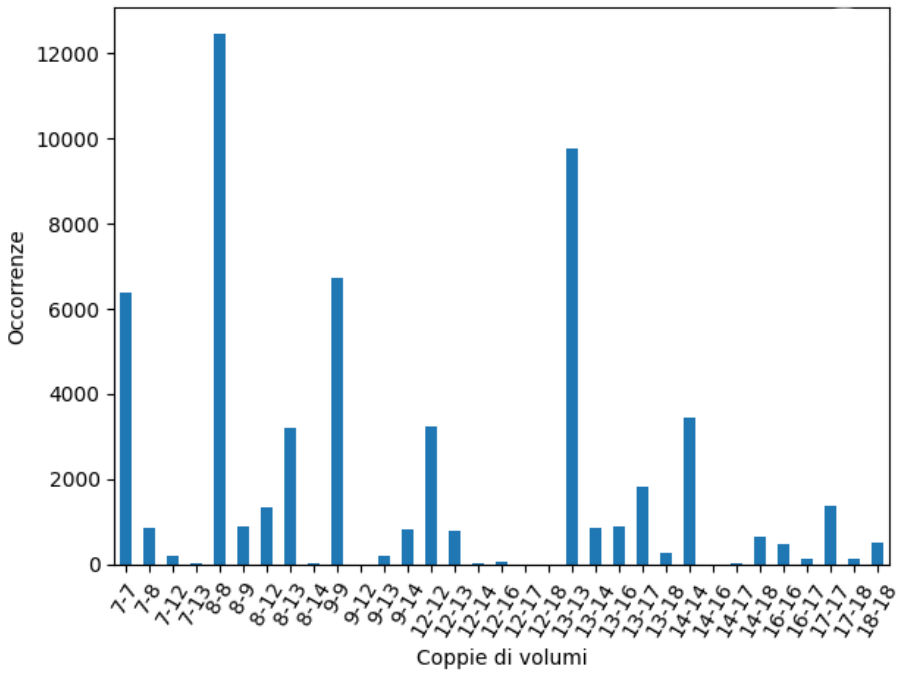

Figura 3.7: Istogramma che mostra le occorrenze dei doublets per coppie di volumi.

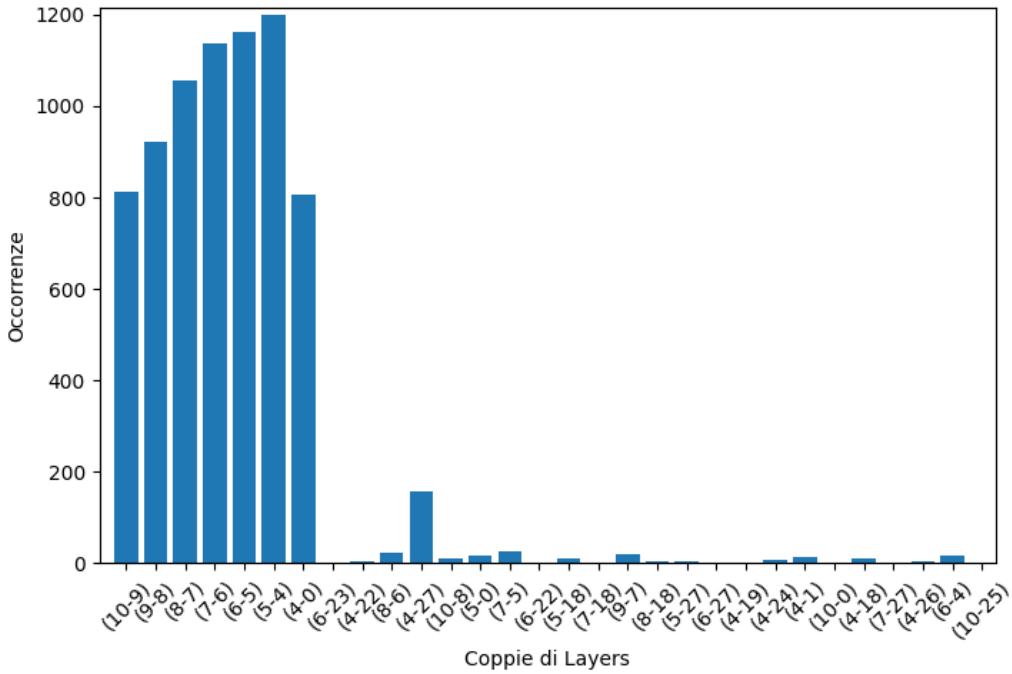

Figura 3.8: Istogramma che mostra le occorrenze dei doublets che contengono hit registrati nel volume 7. Sono state scartate le combinazioni di layer contenenti un numero di occorrenze inferiore a 20.

layer sono invece più rari (basso numero di occorrenze); pertanto, si possono trascurare nella fase di ricostruzione.

3.6 Ricostruzione delle traiettorie con Patatrack

Perché sia possibile utilizzare il software Patatrack per risolvere il problema posto dalla TrackML challenge è necessario innanzitutto adattare il codice in modo da leggere e utilizzare correttamente i dati forniti.
La ricostruzione locale (paragrafo 2.3) non è più necessaria. È pero necessario sviluppare un metodo per inserire gli hit nella catena di ricostruzione, per poter essere utilizzati per la ricostruzione delle tracce. Per fare questo è necessario ridefinire la funzione `makeHits`, che usa i dati per riempire i membri della classe `TrackingRecHit2DCPU`, cioè i cluster, ottenendo così un oggetto che contiene tutte le informazioni riguardanti gli hit, come posizione nello spazio tridimensionale e angolo azimutale ϕ.
Successivamente è necessario creare un nuovo `EDProducer`.
Un `EDProducer` ha il compito di creare una collezione di oggetti che possono essere successivamente usati ("consumati") da un `EDProducer` successivo. In questo caso, è stato sviluppato un nuovo `EDProducer` con il compito di creare la nuova collezione di RecHits a partire dai dati in input.
I rechits prodotti vengono successivamente consumati da un secondo `EDProducer`, che ha il compito di produrre gli n-tupletti.
Per semplificare l'implementazione vengono utilizzati soltanto gli hit corrispondenti ai volumi 7, 8 e 9 (la zona blu della Fig. 3.1).
È necessario specificare all'algoritmo per la creazione degli n-tupletti quali coppie di layer debbano essere considerate. Questo puo essere fatto specificando una struttura dati, `layerPairs`, che contiene tutte le coppie di layer che devono essere considerate per la ricostruzione dei doppietti.
Quindi utilizzando le informazioni acquisite nello studio dei doppietti (paragrafo 3.5) si scartano le coppie di layer che producono pochi doppietti, e si inseriscono quelle più rilevanti in `layerPairs` 3.1).

Una volta creati i doppietti, questi vengono combinati per formare degli n-tupletti (2.3.1), che, una volta fittati, costituiscono la traccia. Questo viene fatto considerando l'eventuale allineamento di coppie di doppietti. Una volta ottenuti tutti gli n-tupletti vengono riempiti due strutture dati, `hitIndices` e `detIndices`:

- `hitIndices` contiene gli *hit-id* di tutti gli hit di tutti gli n-tupletti messi in successione.

- `detIndices` contiene le posizioni degli hit iniziali di ogni n-tupletto, messe in successione. Questa informazione puo essere usata per calcolare la

dimensione di ogni n-tupletto (sottraendo la posizione del hit iniziale di un n-tupla con quella del precedente si ottiene il numero di hit contenute nel n-tupletto precedente).

Listing 3.1: Vettore `layerPairs` per la costruzione dei doppietti nella zona blu (Fig. 3.1) del detector TrackML. "BV" indica che si considerano hit del barrel (barrel volume), mentre "DV" indica che si considerano hit di uno dei due disk (disk volume).

```
constexpr uint8_t layerPairs[2 * nPairs] = {
  0, 1, 1, 2, 2, 3                                  // BV8
  4, 5, 5, 6, 6, 7, 7, 8, 8, 9, 9, 10,              // DV7
  11, 12, 12, 13, 13, 14, 14, 15, 15, 16, 16, 17, // DV9
  0, 4, 1, 4,                                       // BV8DV7
  0, 2, 1, 2, 2, 3                                  // BV8DV9
};
```

Una volta adattato il codice, e applicando un taglio sul momento trasverso di $2\,GeV$, Patatrack ha ricostruito 268 traiettorie, mentre le traiettorie derivanti dalla verita sono 253. Una volta costruiti gli n-tupletti questi vengono fittati con delle tecniche che tengono in considerazione lo scattering delle particelle nei rivelatori e la perdita di energia delle particelle. I fit hanno due obiettivi principali: eliminare le traiettorie "fake", con dei cut sul χ^2; e estrapolare dai fit i parametri della traiettoria, che vanno a costituire l'oggetto finale: le tracce.

Capitolo 4

Conclusioni e sviluppi futuri

Dopo aver studiato la geometria del rivelatore su cui si basano i dati della TrackML challenge, sono state applicate delle modifiche al software Patatrack per adattarlo a questa nuova geometria. Le modifiche apportate al software di ricostruzione hanno consentito di ricostruire con successo gli n-tupletti; tuttavia l'adattamento dei fit alla nuova geometria è ancora in fase di sviluppo e porterà alla ricostruzione finale delle tracce.
In futuro sarà poi necessario effettuare uno studio delle performance di fisica ottenute, come efficienza e fake-rate, e di quelle computazionali, che sono a loro volta di estrema importanza per gli algoritmi di ricostruzione negli esperimenti di fisica delle alte energie. Un ulteriore sviluppo futuro consiste nell'estendere l'algoritmo a tutto il rivelatore. Infatti, le traiettorie sono state ricostruite considerando per ora soltanto gli hit corrispondenti a tre dei nove volumi del detector, ed è quindi necessario estendere la ricostruzione anche agli altri volumi. Infine è necessario compiere uno studio dei parametri e dei tagli usati per la ricostruzione, in modo da trovare una scelta ottimale che massimizzi l'efficienza e minimizzi il fake-rate, mantenendo sotto controllo il tempo di esecuzione dell'algoritmo.

Bibliografia

[1] The CMS experiment at the CERN LHC. The Compact Muon Solenoid experiment. *JINST*, 3:S08004. 361 p, 2008. Also published by CERN Geneva in 2010.

[2] Virdee T. S. Beyond the standard model of particle physics. *Royal Society*, 2016.

[3] Rudolf Frühwirth and Are Strandlie. *Pattern Recognition, Tracking and Vertex Reconstruction in Particle Detectors*. Springer Cham, 1 edition, 2021.

[4] A. Bocci, V. Innocente, M. Kortelainen, F. Pantaleo, and M. Rovere. Heterogeneous reconstruction of tracks and primary vertices with the cms pixel tracker. *Frontiers in Big Data*, 3, 2020.

[5] Felice Pantaleo. New Track Seeding Techniques for the CMS Experiment, 2017.

[6] Polo Calafiura, Steven Farrell, Heather Gray, Jean-Roch Vlimant, Vincenzo Innocente, Andreas Salzburger, Sabrina Amrouche, Tobias Golling, Moritz Kiehn, Victor Estrade, Cécile Germaint, Isabelle Guyon, Ed Moyse, David Rousseau, Yetkin Yilmaz, Vladimir Vava Gligorov, Mikhail Hushchyn, and Andrey Ustyuzhanin. Trackml: A high energy physics particle tracking challenge. In *2018 IEEE 14th International Conference on e-Science (e-Science)*, pages 344–344, 2018.

[7] Kaggle. `https://www.kaggle.com/`.

[8] X. Ai. Acts: A common tracking software. *arXiv: Instrumentation and Detectors*, 2019.

[9] The pandas development team. pandas-dev/pandas: Pandas, February 2020.

Ringraziamenti

Voglio iniziare ringraziando il Professor Francesco Giacomini per avermi permesso di partecipare a questo bellissimo progetto, da cui ho imparato tanto. Voglio ringraziare anche Felice e Tony, per il loro aiuto durante tutto il progetto, e un ringraziamento speciale a Wahid, per la sua pazienza infinita e per il suo aiuto, che è stato a dir poco indispensabile.
Ringrazio i miei genitori: mia madre, che ha sempre avuto fiducia nelle mie capacità, e mio padre, la cui stima in me non è mai venuta meno. Voglio ringraziare poi i miei nonni e tutti i miei parenti.
In particolare devo ringraziare mia sorella Milena, che mi ha sempre sostenuto in tutto e mi ha fatto capire quale dono possa essere avere una sorella.
Ringrazio poi i miei amici più cari, Cassin, Danny e Christal, con cui ho condiviso tanti bei momenti negli anni, e con cui spero di condividerne tanti altri, Alessandro, che è stato il mio braccio destro per questi tre anni, e infine Eleonora, che ha sempre creduto in me, fin dall'inizio.
Ringrazio infine tutte le splendide persone che ho conosciuto in questo percorso: Gabriel, Ciava, Francesco, Niccolò e tutti gli altri.