

Analisi di traiettorie indoor per la determinazione di stop points

Giuseppe Maurizio Facchi
mat. 989910

Gregorio Ghidoli
mat. 983827

A.A. 2021-2022

Indice

1	Introduzione	3
2	Concetti generali	3
3	Analisi del dataset	7
4	Individuazione dei punti di fermata	8
4.1	Algoritmo basato su time-space threshold	9
4.2	Algoritmo di clustering di sequenze	10
4.3	Algoritmo basato su un approccio statistico	14
4.4	Eperimenti	17
5	Calcolo dei tempi di osservazione	21
6	Osservazione congiunta dei visitatori	25
7	Query	27

1 Introduzione

Il progetto è volto a familiarizzare con l'analisi di traiettorie, un tipo di dato spazio-temporale. In particolare, sono state fornite tre traiettorie percorse all'interno di un museo da tre diverse persone, con l'obiettivo di:

1. trovare i punti di fermata delle persone;
2. per ogni persona, calcolare per quanto tempo ha osservato ciascuna esibizione del museo, considerando solo i punti di fermata;
3. per ogni esibizione del museo, calcolarne il tempo totale di osservazione, considerando solo i punti di fermata delle persone nei loro pressi;
4. per ogni esibizione del museo, trovare il numero di persone che la osservano congiuntamente nel tempo.

2 Concetti generali

Prima di presentare il lavoro svolto per il progetto, introduciamo i concetti generali che possono aiutare a capire e a trattare il problema proposto.

L'operazione relativa all'estrazione di informazioni che arricchiscono la conoscenza del dominio di un problema prende il nome di *data mining*. Una possibile organizzazione delle operazioni presenti in letteratura nel caso del paradigma del *trajectory data mining*, ovvero l'estrazione di informazioni dalle traiettorie, è specificata nella Figura 1.

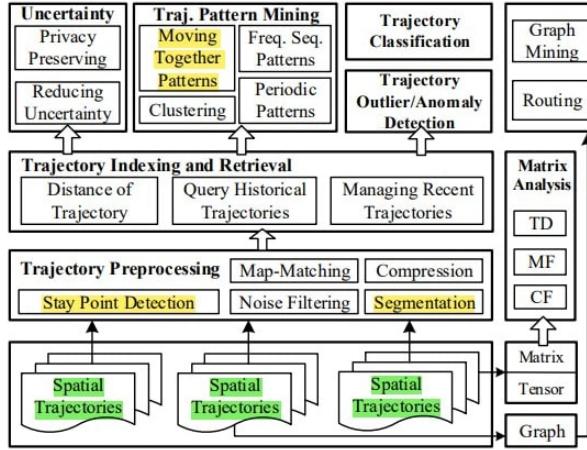


Figura 1: Paradigma del trajectory data mining descritto in [5]. Nella Figura sono evidenziate le fasi affrontate per lo svolgimento del progetto.

Definizione 2.1: Traiettoria

Una traiettoria rappresenta il movimento, ovvero l'evoluzione della posizione nel tempo, di un oggetto nel mondo reale (persona, animale, automobile, ecc...), detto anche *moving object*, in uno spazio geografico durante un dato intervallo di tempo. Tipicamente, una traiettoria viene rappresentata e memorizzata come una sequenza di punti cronologicamente ordinati

$$p_1 \rightarrow p_2 \rightarrow \cdots \rightarrow p_n,$$

dove ogni punto consiste in una coppia di coordinate geografiche associate ad un timestamp

$$p_i = (x_i, y_i, t_i), i = 0, \dots, n, x_i, y_i, t_i \in \mathbb{N}.$$

Durante una traiettoria, il moving object può non cambiare la sua posizione, o cambiarla leggermente, per un determinato periodo di tempo, caratterizzando un punto di fermata.

Definizione 2.2: Punto di fermata - Stop Point

Viene generalmente definito "punto di fermata", un punto nel dominio spazio-temporale dove un'entità risulta essere stazionaria. Esistono tipicamente due condizioni di stazionarietà:

- l'entità rimane ferma nello stesso punto per un certo periodo di tempo;
- definita un'area, l'entità rimane all'interno di essa per un certo periodo di tempo.

Dunque, visualizzando le traiettorie come sequenze di punti, i punti di fermata possono essere identificati da un gran numero di punti sparagliati attorno a una posizione, oppure come un unico punto a cui è assegnato un elevato intervallo di tempo rispetto ai successivi.

Questo comportamento è utile da individuare all'interno di una traiettoria in quanto può rappresentare un'attività importante svolta dal moving object, come fermarsi in un posto per mangiare o, nel caso di questo progetto, visitare un'esibizione di un museo. Da questa osservazione si può quindi pensare di estrarre dalla traiettoria le sequenze di movimento e le sequenze di fermata (*stop*) tramite operazioni di *preprocessing*, come *stay point detection* e *trajectory segmentation*. In questo modo si passa da una visione "piatta" della traiettoria a una visione arricchita di informazioni, ovvero le sequenze di movimento e di fermata, utili per future analisi o processi di *trajectory pattern mining*, come individuare punti di interesse o comportamenti di affollamento.

Definizione 2.3: Sequenza

Una sequenza è qualsiasi serie continua di punti in una traiettoria.

Tipicamente, le sequenze di movimento sono caratterizzate dal fatto che il moving object mantiene una velocità relativamente sostenuta coprendo distanze relativamente lunghe in brevi intervalli di tempo, mentre nelle sequenze di stop mantiene velocità molto basse, anche pari a zero, coprendo distanze molto brevi in intervalli di tempo relativamente lunghi¹.

¹Nelle analisi bisogna fare attenzione a non identificare come sequenze di stop le sequenze in cui il movimento è molto lento ma scorrevole (lineare) o fermate momentanee

Tuttavia, l'inevitabile presenza di rumore all'interno dei dati spazio-temporali relativi a traiettorie complica l'estrazione di sequenze significative e l'analisi di queste ultime. Il rumore è dovuto a errori di misurazione e di campionamento nei sistemi di acquisizione di dati spazio-temporali, chiamati sistemi RLTS, e porta ad avere una deviazione dei punti della traiettoria dalla reale posizione del moving object.

Definizione 2.4: Rumore della traiettoria - Noise

Il rumore della traiettoria (*noise*) è la deviazione dei punti di una traiettoria dalla posizione reale del moving object, dovuta a errori di misurazione e campionamento.

Definizione 2.5: Real-Time Location System - RTLS

Si definisce Real-Time Location System (RTLS) qualsiasi sistema, hardware e software, in grado di campionare accuratamente ad alta frequenza la posizione di un'entità in uno spazio definito.

Gli RTLS forniscono la posizione di un tag e, in base alle loro caratteristiche, una posizione può indicare:

- la presenza in un'area, espressa simbolicamente;
- la posizione precisa, espressa in coordinate;
- la vicinanza ad un altro tag, espressa come distanza o in modo simbolico.

Quando l'entità di cui si acquisisce la posizione partecipa direttamente all'attività di tracciamento tramite l'utilizzo di RTLS, allora si parla di **registrazione attiva**.

Per l'estrazione di sequenze di stop, nel progetto abbiamo codificato e utilizzato tre algoritmi proposti in letteratura e che seguono approcci diversi:

- un algoritmo basato su *time-space threshold* (*PMLH*), presentato in [1];

di brevissima durata.

- un algoritmo di clustering di sequenze (*SOC*) ², presentato in [4];
- un algoritmo basato su un approccio statistico (*MSN*), presentato in [2].

3 Analisi del dataset

Il dataset fornito contiene le traiettorie percorse all'interno di un museo da tre diversi visitatori, la cui posizione è stata registrata attivamente per un intervallo di tempo di circa 10 minuti ad una frequenza di campionamento di circa $12Hz$. In aggiunta, sono stati forniti due shapefile ESRI contenenti rispettivamente le geometrie dei tavoli del museo e le geometrie delle esposizioni. La posizione dei punti tracciati e delle geometrie è rappresentata secondo l'ESPG 3003. Per visualizzare i tavoli, le esibizioni e le traiettorie abbiamo utilizzato QGIS, ottenendo la Figura 2a e la Figura 2b.

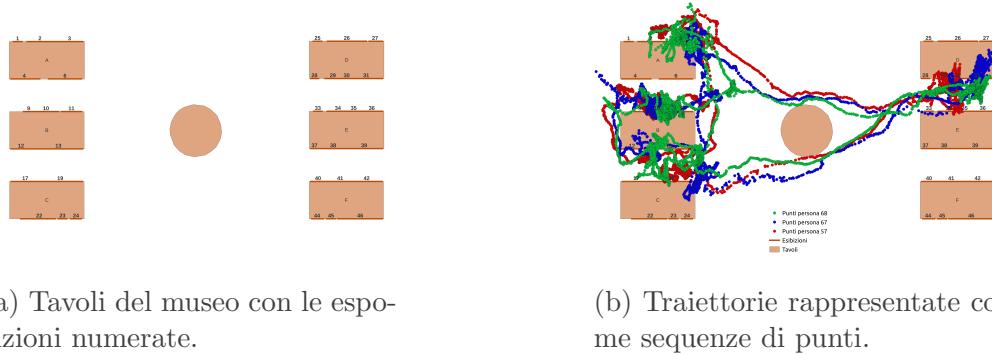


Figura 2

Per ottenere una rappresentazione succinta delle traiettorie abbiamo calcolato le statistiche contenute nella Tabella 1. Ci siamo serviti di questi dati aggregati principalmente per avere una linea guida su come impostare i parametri degli algoritmi utilizzati.

²Sebbene il clustering sia una tecnica di data mining, in questo progetto viene usato come tecnica di preprocessing per l'estrazione di sequenze di stop.

Persona 57

	Distanza	Durata	Velocità
Media	0.0403	0.085	0.4852
Mediana	0.0307	0.072	0.3743
Minimo	0.0002	0.071	0.0022
Massimo	0.5754	0.179	4.1787

Persona 67

	Distanza	Durata	Velocità
Media	0.052	0.089	0.6134
Mediana	0.0389	0.072	0.454
Minimo	0.0006	0.071	0.003
Massimo	0.4939	0.179	6.2258

Persona 68

	Distanza	Durata	Velocità
Media	0.0454	0.088	0.5456
Mediana	0.0343	0.072	0.4169
Minimo	0.0008	0.071	0.0108
Massimo	0.7911	5.356	10.9884

Tabella 1: Statistiche relative alle traiettorie dei visitatori.

4 Individuazione dei punti di fermata

Al giorno d'oggi, grazie alla diffusione di dispositivi come i ricevitori GPS, si hanno a disposizione grandi quantità di dati relativi a traiettorie. Questo tipo di dati è utilizzato in diversi ambiti di ricerca, quali lo studio della mobilità personale, la gestione del trasporto urbano e l'analisi del comportamento umano e animale. Al fine di studiare al meglio questi problemi, è utile individuare accuratamente informazioni semantiche sulle traiettorie originali, come le sequenze di movimento e di fermata. Come descritto in [3], le tecniche per l'estrazione di sequenze di stop presenti in letteratura si dividono in tre classi: metodi basati su *time-distance threshold*, algoritmi di clustering basati sulla densità e metodi basati su approcci statistici. Sono ancora ambito di ricerca i nuovi approcci, ad esempio [3], che considerano gli elementi geografici in prossimità delle traiettorie, tra i quali figurano i *point*

of interest (POI), per un'estrazione delle sequenze più precisa. Per questo progetto abbiamo implementato e sperimentato l'utilizzo di tre algoritmi, ciascuno appartenente a una delle diverse categorie. Dopo aver confrontato i risultati, abbiamo deciso di proseguire le analisi con gli esiti dell'algoritmo di clustering.

4.1 Algoritmo basato su time-space threshold

In [1] viene descritto l'Algoritmo 1, da noi identificato come PMLH, appartenente alla categoria dei metodi basati su time-space threshold. PMLH, infatti, considera solamente la dimensione spaziale e temporale delle traiettorie, senza prendere in considerazione altri eventuali parametri, come la velocità del moving object, o l'inevitabile presenza di rumore nei dati. L'algoritmo identifica sequenze continue di punti che rimangono entro una certa distanza per almeno un periodo di tempo fissato . La funzione *diameter* calcola la distanza più grande tra ogni elemento di un insieme, mentre la funzione *memoid* identifica la posizione del punto in una sequenza che minimizza la massima distanza da ogni altro punto della stessa, ovvero il punto più vicino al centro della sequenza. L'algoritmo restituisce un insieme di tuple $s_i = (seq_i, t_i^{start}, t_i^{end})$. Partendo da questo insieme è facile classificare i punti della traiettoria secondo la loro appartenenza o meno alle fermate identificate dai memoidi seq_i , verificando che il timestamp di un punto sia compreso tra t_i^{start} e t_i^{end} .

Descrizione dei parametri

- Roaming distance, Δl_{roam} : rappresenta la massima distanza a cui un punto p_i può stare da un altro punto p_j per poter essere considerato come facente parte della fermata di p_j ;
- Stay duration, Δt_{dur} : rappresenta la minima durata per cui il moving object deve stare entro la roaming distance di un punto p per poter considerare il punto p come facente parte di una fermata.

Complessità computazionale La complessità computazionale di PMLH è $O(n^2)$, dove n è il numero di punti che compongono la traiettoria.

Algoritmo 1: PMLH

Data: $trajPoints = \{p_1, \dots, p_n\}, \Delta l_{roam}, \Delta t_{dur}$
Risultato: $S = \{s_1, \dots, s_k\}, s_i = (seq_i, t_i^{start}, t_i^{end})$ dove seq_i è una sequenza in $trajPoints$

```
1  $i = 0, S = \emptyset;$ 
2 while  $i < |trajPoints|$  do
3    $j^* = \min j$  s.t.  $p_j \geq p_i + \Delta t_{dur};$ 
4   if  $diameter(i, j^*) > \Delta l_{roam}$  then
5      $i = i + 1;$ 
6   else
7      $j^* = \max j$  s.t.  $diameter(i, j) \leq \Delta l_{roam};$ 
8      $S = S \cup (medoid(i, j^*), t_i, t_{j^*});$ 
9      $i = j^* + 1;$ 
10  end
11 end
```

4.2 Algoritmo di clustering di sequenze

In letteratura sono presenti diversi metodi per la segmentazione di traiettorie che identificano i punti o le sequenze di stop facendo considerazioni sul cambio di profilo della velocità e della direzione del moving object quando questo passa da uno stato di movimento a uno stato di fermata. Un'altro approccio per affrontare il problema consiste nell'uso di tecniche di data-mining, come gli algoritmi di clustering basati sulla densità dove, tra quelli specializzati per i dati spazio-temporali, si annoverano CB-SMoT, T-OPTICS e TrajDB-SCAN. Tuttavia, questi approcci non forniscono buoni risultati quando si ha un elevato rumore nei dati e, inoltre, non determinano come fermate i singoli punti, chiamati *big point*, a cui è associato un elevato intervallo di tempo. Un buon algoritmo di clustering deve ridurre il più possibile la creazione di veri negativi, come le sequenze di stop non individuate o le sequenze di stop separate dal rumore, e il numero di falsi positivi, come le sequenze di movimento molto lento e di forma lineare, cercando quindi di incrementare il rapporto di sequenze di stop effettive individuate. A questo proposito è stato proposto l'algoritmo Sequence Oriented Clustering (SOC), descritto in [4], che permette di determinare i punti di rumore come facenti parte delle sequenze di stop, queste avendo forma arbitraria, e di individuare i big point.

SOC è basato su altri due algoritmi di clustering basati sulla densità: DBSCAN e OPTICS, dove quest'ultimo è una generalizzazione del primo. DBSCAN, come la sua estensione ST-DBSCAN per il clustering di dati spazio-temporali, forma i cluster basandosi sulla nozione di *density-reachability*. Un punto p è chiamato *core-point* se il vicinato di p contiene un numero $minPts$ di altri punti entro un dato raggio eps . Un punto q si dice *directly density-reachable* da p se q è nel vicinato di p . Un punto q si dice *density-reachable* da p se q è collegato a p da una catena di punti tale per cui ciascun punto è *directly density-reachable* al successivo. DBSCAN scansiona ciascun punto p una volta e, se p è un core-point, viene creato un nuovo cluster. Il cluster viene poi espanso ricorsivamente aggiungendo i punti density-reachable dai punti già appartenenti al cluster.

Mentre DBSCAN si basa su singoli core-point, l'algoritmo SOC è orientato al clustering di sequenze. Dall'osservazione che una fermata è costituita da punti tra loro vicini nello spazio e che questi coprono un intervallo temporale relativamente lungo, SOC estrae le sequenze di fermata all'interno di una traiettoria basandosi sul concetto di *core-sequence*: sequenza in cui il moving object rimane in un'area piccola per un lungo periodo di tempo. Una *eps-sequence* di un punto p è la massima sequenza che contiene p e in cui la distanza (euclidea) tra p e tutti gli altri punti della sequenza è minore di un valore eps fissato: eps è il raggio di un'area circolare. Una *core-sequence* è una sequenza che contiene un punto p per cui la sequenza è una *eps-sequence* di p e l'intervallo temporale della sequenza è maggiore o uguale a un valore tau fissato: quindi il clustering è basato su una densità che tiene conto sia della prossimità spaziale che della continuità e durata nel tempo. Un *core-point* è un punto per cui la sua *eps-sequence* è una *core-sequence*. La *core-distance* di un punto p è il minimo raggio r minore di eps tale per cui la *r-sequence* è una *core-sequence* se la *eps-sequence* di p è una *core-sequence*, altrimenti è un valore *UNDEFINED* predefinito: rappresenta la vicinanza spaziale di un punto ai suoi vicini considerando il tempo, ad esempio è zero nel caso di un big point. La *reachability-distance* di un punto p è calcolata come segue:

1. si controllano p e i punti che precedono p ;
2. si trova il punto q precedente e più prossimo a p nella traiettoria tale per cui p appartiene alla *core-sequence* di q ;

3. se q esiste, allora la reachability-distance è il massimo tra la core distance di q e la distanza tra p e q ;
4. se non esiste, allora la reachability-distance è UNDEFINED.

Una *eps-reachability sequence* è una sequenza di punti tale per cui la reachability distance di ciascun punto è minore o uguale di eps e la reachability-distance del punto che precede la sequenza nella traiettoria e del punto che segue la sequenza nella traiettoria è maggiore di eps .

Le *eps-reachability sequence* rappresentano quindi sequenze che si sovrappongono o si incontrano nel tempo. Una volta calcolata la reachability-distance di ciascun punto, si possono estrarre le *eps-reachability sequence* e considerarle come sequenze di stop. Tuttavia, alcune sequenze di stop successive possono essere state erroneamente interrotte da punti di noise. Per far fronte a questo problema vengono usati due criteri per confrontare *eps-reachability sequence* successive e nel caso unirle:

1. il primo permette di unire due *eps-reachability sequence* se queste sono vicine in spazio e separate da un periodo di tempo troppo breve per essere considerato un tempo di movimento ragionevole: si usa un parametro *minMov* che indica la durata minima di un movimento normale.
2. il secondo permette di unire due *eps-reachability sequence* se i convex-hull delle stesse si intersecano e se sono separate da un periodo di tempo troppo breve per essere considerato un tempo di movimento ragionevole.

Applicando questi criteri, una *eps-reachability sequence* cresce fino a quando nessun'altra sequenza può essere unita ³. Tali sequenze vengono chiamate *Ful-reachability sequence*. Le *Ful-reachability sequence* possono terminare con punti che non fanno parte della sequenza di stop ma che sono raggiungibili dall'ultimo core-point della *Ful-reachability sequence*. In questo caso la *Ful-reachability sequence* viene potata facendola terminare con un core-point e ottenendo così una sequenza di stop. SOC è in grado di generare sequenze di stop, quest'ultime di forma arbitraria, tra cui sequenze di forma lineare che non rappresentano una fermata effettiva ma un movimento lento.

³Quando si uniscono due *eps-reachability sequence* si aggiungono alla sequenza anche i punti di noise intermedi. La reachability distance di questi nodi viene aggiornata al valore eps .

Come ultimo passo dell'algoritmo, vengono quindi filtrate le sequenze di stop false positive per cui gli indici di *straightness* e *centered-distance* presentati in [4] superano una certa soglia.

Descrizione dei parametri

- eps rappresenta il raggio per definire una core-sequence;
- tau rappresenta la durata minima per definire una core-sequence;
- UNDEFINED rappresenta un valore di default per la reachability distance dei punti della traiettoria: seguendo quanto descritto in [4] viene impostato a $1.2 \times \text{eps}$;
- minMov rappresenta la minima durata che un movimento deve avere per essere considerato un movimento normale: viene usato per unire le eps-reachability sequence;
- le misure di straightness e di centered-distance vengono usate per filtrare i falsi positivi: seguendo quanto descritto in [4] vengono impostati ai valori di default, rispettivamente 0.5 e $2 \times \text{eps}$.

La struttura dei cluster è influenzata dal rapporto $\frac{\text{eps}}{\text{tau}}$. Diminuendo il valore di eps si restringono le condizioni per identificare le core-sequence, quindi le eps-reachability sequence diminuiscono in dimensione e in numero. Per questo motivo è più probabile ottenere meno sequenze false positive ma più sequenze vere negative. Diminuendo il valore di tau si rilassano le condizioni per identificare le core-sequence, quindi le eps-reachability sequence aumentano in dimensione e in numero. Per questo motivo è più probabile ottenere più sequenze false positive ma meno sequenze vere negative. Tuttavia, dai risultati degli esperimenti in [4], risulta che eps abbia maggiore influenza di tau sull'identificazione delle sequenze e viene consigliato di impostare questi parametri in modo tale che il rapporto $2 \times \frac{\text{eps}}{\text{tau}}$ assuma valori relativamente piccoli. Infine, diminuendo il valore di minMov si diminuisce la capacità di unire le eps-reachability sequence. Il parametro minMov deve essere grande abbastanza da eliminare l'influenza del rumore nelle traiettorie, ma non troppo grande per evitare di identificare movimenti curvi come sequenze di stop. In generale, il valore di minMov deve essere incrementato quando si ha un elevato numero di punti con rumore.

Complessità computazionale La complessità di SOC, come per DBSCAN e OPTICS, è $O(n * \log n)$, , dove n è il numero di punti che compongono la traiettoria.

Algoritmo 2: SOC

Data: $trajPoints, eps, tau, \text{UNDEFINED}, minMov$

Result: $stops = \{s_1, \dots, s_k\}$ dove s_i è una sequenza in $trajPoints$

```

1  $\forall p \in trajPoints$   $p.\text{reachability distance} = \text{UNDEFINED};$ 
2 for  $i = 1$  to  $|trajPoints|$  do
3    $seq = \text{epsSequence}(trajPoints, p_i, eps);$ 
4   if  $\text{isCoreSequence}(seq, tau)$  then
5      $r = \text{coreDistance}(seq);$ 
6     for  $j = i$  to  $|trajPoints|$  do
7        $| p_j.\text{reachability distance} = \max\{r, \text{distance}(p_i, p_j)\}$ 
8     end
9   end
10 end
11  $\text{epsReachabilitySeqs} = \text{extractSequences}();$ 
12  $stops = \text{mergeSequences}(minMov);$ 
13  $\text{pruneStops}(stops);$ 
14  $\text{falsePositiveRemove}(stops);$ 

```

4.3 Algoritmo basato su un approccio statistico

In [1] viene trattata la segmentazione di traiettorie in segmenti di movimento, di fermata e di rumore, come un problema di identificazione di outlier. Per l'identificazione degli outlier in serie temporali, gli autori utilizzano la statistica *modified z-score*, rappresentata nell'Equazione 1. Le costanti 0.6745 e 1.4826 servono per rendere le rispettive statistiche non deviate rispetto alla distribuzione normale. La scelta di utilizzare il modified z-score è motivata dalle scarse prestazioni delle misure di media e deviazione standard in presenza di outlier.

$$M_i = \frac{0.6745(x_i - \bar{x})}{\text{MAD}}, \text{ con } \text{MAD} = 1.4826 \times \text{median}(Y_i - \tilde{Y}) \quad (1)$$

Nell'articolo viene suggerito di trattare come outlier i punti che hanno un modified z-score in valore assoluto più grande di 3.5 e viene proposto l'Algo-

ritmo MSN 3 per la segmentazione di traiettorie in segmenti di movimento, di fermata e di rumore. Per prima cosa, MSN rileva potenziali punti di rumore che hanno distanze relativamente lunghe. Successivamente, assume che un singolo angolo acuto nella traiettoria rappresenti un movimento di “tornare indietro”, mentre due angoli acuti consecutivi siano da considerare potenziali segmenti di rumore. Infine, dopo aver rimosso i punti di rumore, identifica i punti la cui durata è molto lunga e la velocità molto bassa ⁴, classificandoli come punti di stop ⁵.

Descrizione dei parametri

- $S_\tau, T_\tau, V_\tau, A_\tau$: rappresentano le serie di distanza, durata, velocità e angolo di sterzata, relative ai punti della traiettoria τ ;
- $\epsilon_s, \epsilon_t, \epsilon_v$: rappresentano le soglie del modified z-score rispettivamente per la distanza, la durata e la velocità;
- θ : rappresenta l’angolo minimo di sterzata, usato per migliorare l’identificazione del rumore;
- ρ : rappresenta i limiti dell’intervallo $[-\rho, \rho]$ da cui vengono estratti casualmente in modo uniforme i valori di jitter da aggiungere ai valori di T_τ per evitare di avere $MAD = 0$.

I valori dei parametri consigliati dagli autori sono: $\epsilon_s = 3.5, \epsilon_t = 3.5, \epsilon_v = 3.5, \theta = 45, \rho = 0.5$.

Complessità computazionale La complessità di MSN è $O(n)$, dove n è il numero di punti che compongono la traiettoria.

⁴L’indice modified z-score dovrebbe essere applicato a dati normalmente distribuiti. Data la natura sbilanciata delle velocità, questa viene normalizzata applicando il logaritmo naturale al fine di ristabilire la simmetria.

⁵L’algoritmo non tiene conto che punti successivi con velocità inferiore alla soglia specificata possono rappresentare sequenze di movimento e non sequenze di fermata, ottenendo quindi dei falsi-positivi.

Algoritmo 3: MSN

Data: $S_\tau, T_\tau, V_\tau, A_\tau, \epsilon_s, \epsilon_t, \epsilon_v, \theta, \rho$
Result: $move_indexes, stop_indexes, noise_indexes$

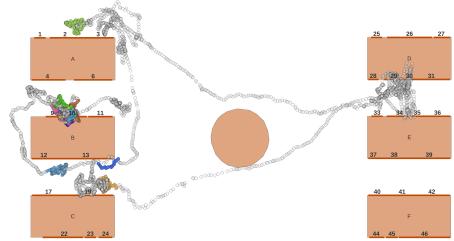
```
1  $distance\_outliers = []$ ,  $direction\_outliers = []$ ,  $duration\_outliers = []$ ;  
2  $M_s = modifiedZScore(S_\tau, MAD_s, \bar{s})$ ;  
3 for  $i = 0$  to  $length(M_s)$  do  
4   if  $M_s[i] > \epsilon_s$  then  
5     | Appendi  $i$  a  $distance\_outliers$ ;  
6   end  
7 end  
8 for  $i = 0$  to  $length(A_\tau)$  do  
9   if  $A_\tau[i] < \theta$  and  $A_\tau[i + 1] < \theta$  then  
10    | Appendi  $i$  e  $i + 1$  a  $direction\_outliers$ ;  
11    |  $i++$   
12  end  
13 end  
14  $noise\_indexes = distance\_outliers \cup direction\_outliers$ ;  
15  $clean\_indexes = \tau \setminus \tau[note\_indexes]$ ;  
16  $\tau = \tau[clean\_indexes]$ ;  
17  $T_\tau = T_\tau + \rho$ ;  
18  $M_t = modifiedZScore(T_\tau, MAD_t, \bar{t})$ ;  
19 for  $i = 0$  to  $length(M_t)$  do  
20   if  $M_t[i] > \epsilon_t$  then  
21     | Appendi  $i$  a  $duration\_outliers$ ;  
22   end  
23 end  
24  $V_\tau = ln V_\tau$ ;  
25  $speed\_outliers = []$ ;  
26  $M_v = modifiedZScore(V_\tau, MAD_v, \bar{v})$ ;  
27 for  $i = 0$  to  $length(M_v)$  do  
28   if  $M_v[i] < -\epsilon_v$  then  
29     | Appendi  $i$  a  $speed\_outliers$ ;  
30   end  
31 end  
32  $stop\_indexes = duration\_outliers \cap speed\_outliers$ ;  
33  $move\_indexes = clean\_indexes \setminus stop\_outliers$ ;
```

4.4 Esperimenti

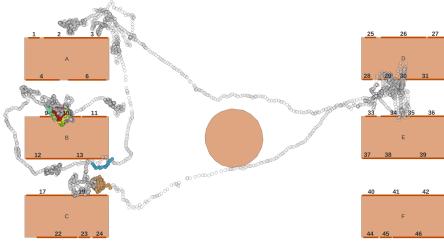
Come accennato in [3], la difficoltà principale negli algoritmi descritti precedentemente risiede nell'impostare correttamente i valori dei parametri. Inoltre, i valori di default proposti negli articoli citati fanno riferimento a punti registrati da ricevitori GPS in ambienti outdoor e alcuni di questi vanno quindi adattati al contesto di questo progetto. Analizzando la Figura 2b si vede come le possibili fermate siano costituite da punti racchiusi in aree circolari con raggio di circa 1 metro. Per questo motivo, in accordo con la Definizione 2, abbiamo impostato a 1 i parametri relativi all'area da considerare per identificare eventuali punti di fermata. In particolare, abbiamo eseguito i seguenti esperimenti:

- esecuzione dell'algoritmo PMLH impostando il parametro $\Delta l_{roam} = 1$ e il parametro $\Delta t_{dur} \in \{20, 30\}$, ottenendo i risultati in Figura 3;
- esecuzione dell'algoritmo SOC impostando il parametro $eps = 1$, il parametro $tau \in \{20, 30\}$ e il parametro $minMov = 0.5$, ottenendo i risultati in Figura 4;
- esecuzione dell'algoritmo MSN impostando il parametro $\epsilon_s = 3 * \sigma_{MS_\tau}$, il parametro $\epsilon_t = \sigma_{MT_\tau}$, il parametro $\epsilon_v = \sigma_{MV_\tau}$, il parametro $\theta = 45$ e il parametro $\rho = 0.0001$, ottenendo i risultati in Figura 5.

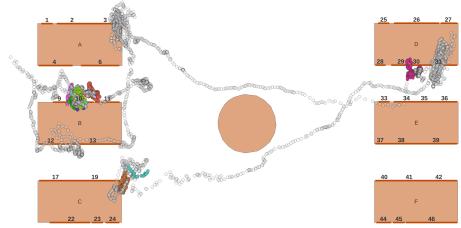
PMLH, oltre ad avere una complessità computazionale maggiore di SOC e di MSN, restituisce risultati meno precisi in quanto non tratta minimamente il problema della presenza di rumore nei dati: identifica infatti molte più fermate di SOC, tra le quale ve ne sono molte false positive. MSN invece, secondo [2], non è un algoritmo congeniale nel caso in cui i dati vengono campionati a frequenza regolare e non fornisce un risultato adatto per le successive analisi: esegue infatti una segmentazione delle traiettorie e non una classificazione dei punti. Infine SOC, oltre ad avere una buona complessità computazionale, da riscontro grafico sembra individuare sequenze di stop sensate. Per questi motivi, abbiamo deciso di proseguire le analisi con i risultati forniti dall'algoritmo SOC eseguito con $tau = 20$.



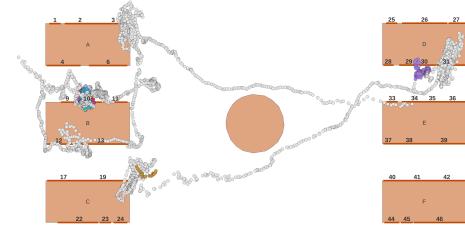
(a) Parametri: $\Delta l_{roam}=1$, $\Delta t_{dur}=20$.
Fermate individuate: 19.



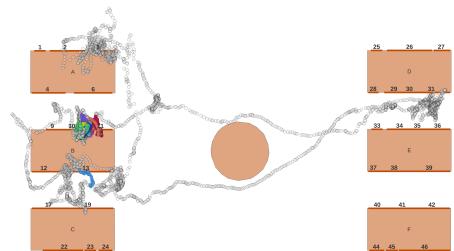
(b) Parametri: $\Delta l_{roam}=1$, $\Delta t_{dur}=30$.
Fermate individuate: 11.



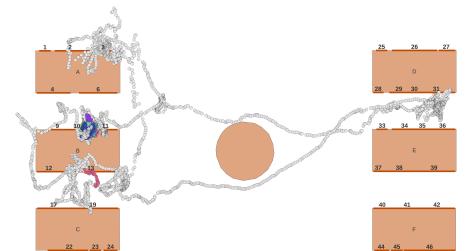
(c) Parametri: $\Delta l_{roam}=1$, $\Delta t_{dur}=20$.
Fermate individuate: 17.



(d) Parametri: $\Delta l_{roam}=1$, $\Delta t_{dur}=30$.
Fermate individuate: 8.

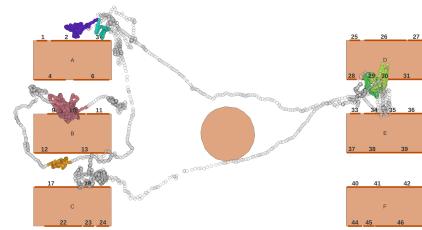


(e) Parametri: $\Delta l_{roam}=1$, $\Delta t_{dur}=20$.
Fermate individuate: 14.

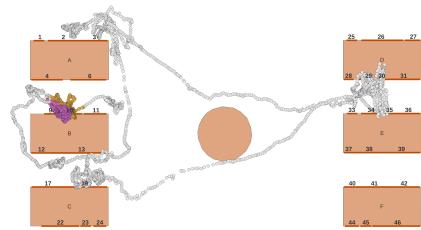


(f) Parametri: $\Delta l_{roam}=1$, $\Delta t_{dur}=30$.
Fermate individuate: 8.

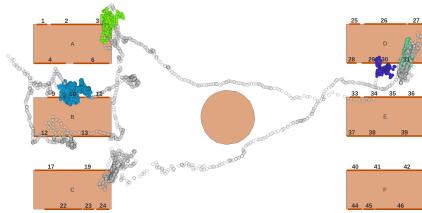
Figura 3: Punti di stop della persona 57 (a,b), della persona 67 (c,d) e della persona 68 (e,f), individuati con l'algoritmo PMLH.



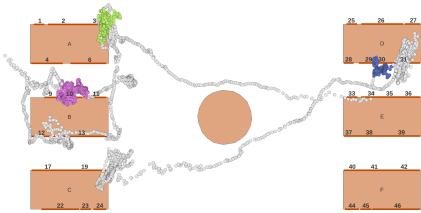
(a) Parametri: $\text{eps}=1$, $\text{tau}=20$, $\text{min-Mov}=0.5$. Fermate individuate: 6.



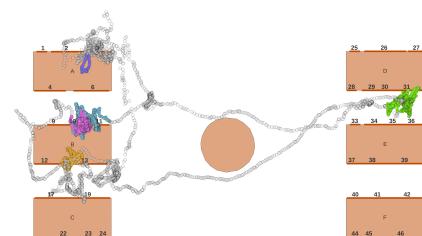
(b) Parametri: $\text{eps}=1$, $\text{tau}=30$, $\text{min-Mov}=0.5$. Fermate individuate: 2.



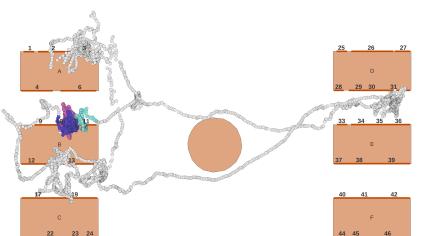
(c) Parametri: $\text{eps}=1$, $\tau=20$, $\text{min-Mov}=0.5$. Fermate individuate: 5.



(d) Parametri: $\text{eps}=1$, $\text{tau}=30$, $\text{min-Mov}=0.5$. Fermate individuate: 3.



(e) Parametri: $\text{eps}=1$, $\text{tau}=20$, $\text{min-Mov}=0.5$. Fermate individuate: 5.



(f) Parametri: $\text{eps}=1$, $\text{tau}=30$, $\text{min-Mov}=0.5$. Fermate individuate: 3.

Figura 4: Punti di stop della persona 57 (a,b), della persona 67 (c,d) e della persona 68 (e,f), individuati con l'algoritmo SOC.

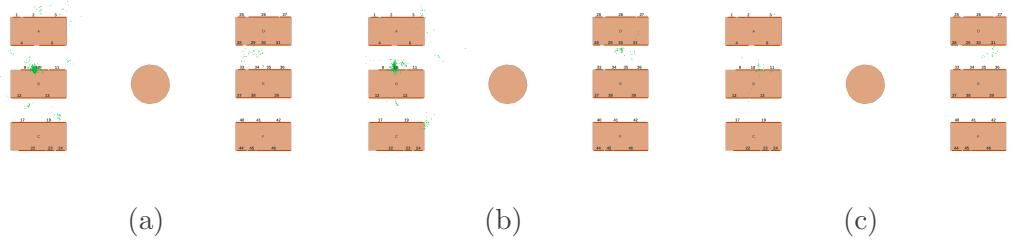


Figura 5: Segmenti di stop della persona 57 (a), della persona 67 (b) e della persona 68 (c), individuati con l'algoritmo MSN.

5 Calcolo dei tempi di osservazione

Parte delle analisi eseguite sui risultati dell'algoritmo SOC riguardano il calcolo dei seguenti tempi di osservazione:

1. per ogni persona, calcolare per quanto tempo ha osservato ciascuna esibizione del museo, tenendo conto dei soli punti di fermata;
2. per ogni esibizione del museo, calcolare il tempo totale di osservazione delle persone nei suoi pressi, tenendo conto dei soli punti di fermata.

Nelle analisi abbiamo tenuto conto delle seguenti assunzioni:

- un visitatore osserva un'esibizione se si trova a una distanza di al più 0.7 metri dalla stessa;
- un visitatore, dopo aver visitato un'esibizione, non torna a visitarla in un secondo momento.

Come prima cosa abbiamo creato dei buffer di raggio 0.7 metri intorno alle geometrie delle esibizioni, come mostrato in Figura 6, per poter tenere nelle analisi solo dei punti che intersecano tali buffer. Successivamente, abbiamo disambiguato l'esibizione a cui fanno riferimento i punti che intersecano due o più esibizioni, considerando l'esibizione che ha distanza punto-centroide dell'esibizione più corta. Poi, abbiamo calcolato i tempi di visita relativi a ciascun visitatore e i tempi totali di visita delle esibizioni, ottenendo i risultati in Tabella 2 e in Tabella 4. Infine, per tener conto anche della presenza di rumore nei dati, abbiamo ripetuto le analisi considerando tutti i punti di tutti i cluster i cui centroidi intersecano i buffer delle esibizioni, ottenendo i risultati in Tabella 3 e in Tabella 5.

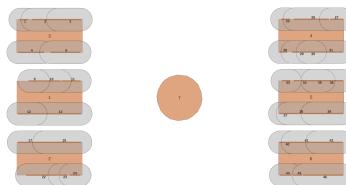


Figura 6: Buffer di raggio 0.7 delle esibizioni.

Visitatore	Esibizione	Tempo di visita
57	2.00	18.75
57	3.00	15.07
57	9.00	39.20
57	10.00	284.75
57	11.00	0.07
57	12.00	15.25
57	13.00	4.82
57	29.00	24.07
57	30.00	20.74
67	3.00	37.24
67	9	5.85
67	10	310.78
67	11	8.25
67	29	2.86
67	30	30.53
67	31	17.10
68	3	2.43
68	6	0.07
68	10	204.66
68	11	116.69
68	12	0.86
68	13	17.46
68	30	2.32
68	31	45.70
68	35	0.57
68	36	4.14

Tabella 2: Tempo di visita delle esibizioni per ciascun visitatore.

Visitatore	Esibizione	Tempo di visita
57	2.00	24.17
57	3.00	7.93
57	10.00	330.42
57	12.00	20.67
57	29.00	25.32
57	30.00	20.74
67	3.00	37.24
67	10.00	327.35
67	30.00	34.74
67	31.00	17.10
68	3.00	2.43
68	10.00	189.06
68	11.00	141.29
68	13	18.75
68	31	67.13
67	31	17.10
68	3	2.43
68	6	0.07
68	10	204.66
68	11	116.69
68	12	0.86
68	13	17.46
68	30	2.32
68	31	45.70
68	35	0.57
68	36	4.14

Tabella 3: Tempo di visita delle esibizioni per ciascun visitatore, considerando tutti i punti dei cluster i cui centroidi intersecano i buffer delle esibizioni.

Esibizione	Tempo di visita
2	18.75
3	54.74
6	0.07
9	45.06
10	800.19
11	125.00
12	16.10
13	22.28
29	26.92
30	53.59
31	62.81
35	0.57
36	4.14

Tabella 4: Tempo totale di visita delle esibizioni.

Esibizione	Tempo di visita
2	24.17
3	47.60
10	846.82
11	141.29
12	20.67
13	18.75
29	25.32
30	55.49
31	84.23

Tabella 5: Tempo totale di visita delle esibizioni, considerando tutti i punti dei cluster i cui centroidi intersecano i buffer delle esibizioni.

6 Osservazione congiunta dei visitatori

L’analisi si conclude con il trovare, per ogni esibizione del museo, il numero di persone che la osservano congiuntamente nel tempo. Per rispondere a questo quesito ci siamo serviti di MobilityDB [6], una piattaforma per la gestione e l’analisi di traiettorie geospaziali. MobilityDB estende PostgreSQL e PostGIS con tipi di dato temporali (*ttype*) e geometrie temporali (*tgeom*) che semplificano l’analisi delle traiettorie. Dai punti disambiguati che intersecano i buffer delle esibizioni abbiamo creato le geometrie temporali che rappresentano i punti di fermata davanti alle esibizioni di ciascun visitatore. Su queste geometrie abbiamo poi applicato la funzione *tcount*, la quale esegue l’interpolazione *stepwise* per descrivere l’evoluzione temporale dei valori. Nel nostro caso, viene restituito un *tint* che rappresenta la variazione nel tempo del numero di visitatori che osservano l’esibizione (Figura 7). La struttura delle tabelle utilizzate, la query eseguita e il diagramma a blocchi riassuntivo sono descritti nella Sezione 7. Un esempio di *tint* è data dalla sequenza temporale:

```
[1@2021-12-13 12:08:52.424+01, 2@2021-12-13 12:08:55.537+01, 3@2021-12-13 12:09:25.787+01, 3@2021-12-13 12:14:20.27+01], (2@2021-12-13 12:14:20.27+01, 2@2021-12-13 12:14:20.349+01], (1@2021-12-13 12:14:20.349+01, 1@2021-12-13 12:14:24.962+01].
```

Il valore di una sequenza temporale è interpretato assumendo che l’intervallo di tempo definito da ciascuna coppia di valori consecutivi sia chiuso a sinistra e aperto a destra, a meno che non siano la prima o l’ultima istanza della sequenza: in quel caso si applica il tipo di intervallo dell’intera sequenza. Quindi la precedente sequenza viene interpretata nel seguente modo:

- l’esibizione è osservata da 1 visitatore dall’istante 2021-12-13 12:08:52.424+01 all’istante precedente a 2021-12-13 12:08:55.537+01;
- l’esibizione è osservata da 2 visitatori dall’istante 2021-12-13 12:08:55.537+01 all’istante precedente a 2021-12-13 12:09:25.787+01;
- l’esibizione è osservata da 3 visitatori dall’istante 2021-12-13 12:09:25.787+01 all’istante 2021-12-13 12:14:20.27+01;
- l’esibizione ritorna ad essere visitata da 2 visitatori dall’istante successivo a 2021-12-13 12:14:20.27+01 all’istante 2021-12-13 12:14:20.349+01.

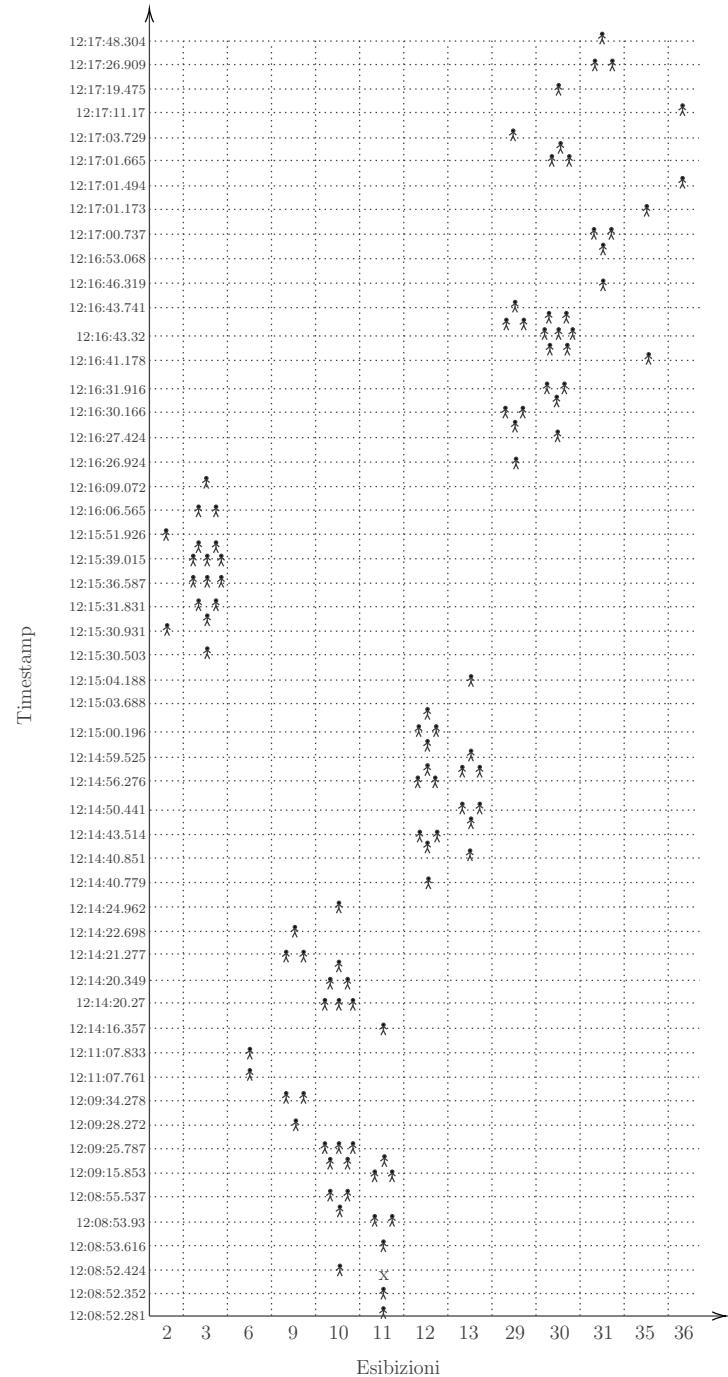


Figura 7: Variazione nel tempo delle persone che osservano le esibizioni.

7 Query

La seguente query calcola quante persone vedono ciascuna esibizione nel tempo. Le tabelle utilizzate sono:

- `stop_points_p{57,67,68}_SOC_eps1_tau20_minMov05`: contengono i punti classificati utilizzando l'algoritmo SOC. La colonna *Segment* contiene il valore “move” se il punto appartiene a una sequenza di movimento oppure “stop_{indiceFermata}” se appartiene a una sequenza di stop.
 - id:Integer
 - geom:Geometry
 - x:Double Precision
 - y:Double Precision
 - timestamp:Varchar
 - Segment:Varchar
- `exhibits_on_tables_buffers_07m`: contiene le geometrie dei buffer di raggio 70cm costruiti intorno alle esibizioni.
 - id:Integer
 - geom:Geometry

```
1 WITH sp57_min_distance_from_buffers AS (
2     SELECT sp57.id , min(st_distance(sp57.geom,
3         st_centroid(eb.geom))) as min_distance
4     FROM "stop_points_p57_SOC_eps1_tau20_minMov05"
5     AS sp57
6         JOIN exhibits_on_tables_buffers_07m AS eb ON
7             st_intersects(sp57.geom, eb.geom)
8             WHERE sp57.segment != 'move'
9             GROUP BY sp57.id
10    sp57_stop_points_in_nearest_exh_buffers AS (
11        SELECT 57 AS person_id , sp57.id as point_id ,
12        sp57.geom, sp57.timestamp , eb.id as exh_id , st_distance(sp57.
13        geom, st_centroid(eb.geom)) as distance_from_centroid
14        FROM "stop_points_p57_SOC_eps1_tau20_minMov05"
15    AS sp57
```

```

11      JOIN exhibits_on_tables_buffers_07m AS eb ON
12      st_intersects(sp57.geom, eb.geom)
13      JOIN sp57_min_distance_from_buffers AS
14      sp57_min_dist ON sp57.id = sp57_min_dist.id
15      WHERE sp57_min_dist.min_distance = st_distance(
16          sp57.geom, st_centroid(eb.geom))
17      sp67_min_distance_from_buffers AS (
18          SELECT sp67.id , min(st_distance(sp67.geom,
19              st_centroid(eb.geom))) as min_distance
20          FROM "stop_points_p67_SOC_eps1_tau20_minMov05"
21          AS sp67
22          JOIN exhibits_on_tables_buffers_07m AS eb ON
23          st_intersects(sp67.geom, eb.geom)
24          WHERE sp67.segment != 'move'
25          GROUP BY sp67.id
26          sp67_stop_points_in_nearest_exh_buffers AS (
27              SELECT 67 as person_id, sp67.id as point_id,
28                  sp67.timestamp, eb.id as exh_id, st_distance(sp67.
29                      geom, st_centroid(eb.geom)) as distance_from_centroid
30              FROM "stop_points_p67_SOC_eps1_tau20_minMov05"
31              AS sp67
32              JOIN exhibits_on_tables_buffers_07m AS eb ON
33              st_intersects(sp67.geom, eb.geom)
34              WHERE sp67.segment != 'move'
35              GROUP BY sp67.id
36              sp68_stop_points_in_nearest_exh_buffers AS (
37                  SELECT 68 as person_id, sp68.id as point_id,
38                      sp68.timestamp, eb.id as exh_id, st_distance(sp68.
39                          geom, st_centroid(eb.geom)) as distance_from_centroid
40                  FROM "stop_points_p68_SOC_eps1_tau20_minMov05"
41                  AS sp68
42                  JOIN exhibits_on_tables_buffers_07m AS eb ON
43                  st_intersects(sp68.geom, eb.geom)
44                  WHERE sp68.segment != 'move'
45                  GROUP BY sp68.id
46                  sp68_stop_points_in_nearest_exh_buffers AS (
47                      SELECT 68 as person_id, sp68.id as point_id,
48                          sp68.timestamp, eb.id as exh_id, st_distance(sp68.
49                              geom, st_centroid(eb.geom)) as distance_from_centroid
50                      FROM "stop_points_p68_SOC_eps1_tau20_minMov05"
51                      AS sp68
52                      JOIN exhibits_on_tables_buffers_07m AS eb ON
53                      st_intersects(sp68.geom, eb.geom)
54                      JOIN sp68_min_distance_from_buffers AS

```

```

37     sp68_min_dist ON sp68.id = sp68_min_dist.id
38     WHERE sp68_min_dist.min_distance = st_distance(
39         sp68.geom, st_centroid(eb.geom))
40     person_tstop_points AS (
41         SELECT person_id, exh_id, tgeompoin_seq(array_agg(
42             tgeompoin_inst(geom, timestamp) ORDER BY timestamp)) AS
43             tpoint
44     FROM (
45         SELECT * FROM sp57_stop_points_in_nearest_exh_buffers
46         UNION
47         SELECT * FROM sp67_stop_points_in_nearest_exh_buffers
48         UNION
49         SELECT * FROM sp68_stop_points_in_nearest_exh_buffers
50     ) AS stop_points_in_nearest_exh_buffers
51     GROUP BY person_id, exh_id
52     exhibit_persons_over_time AS (
53         SELECT exh_id, tcount(tpoint) as persons_over_time
54         FROM person_tstop_points
55         GROUP BY exh_id
56         ORDER BY exh_id
57
58
59     SELECT * FROM exhibit_persons_over_time

```

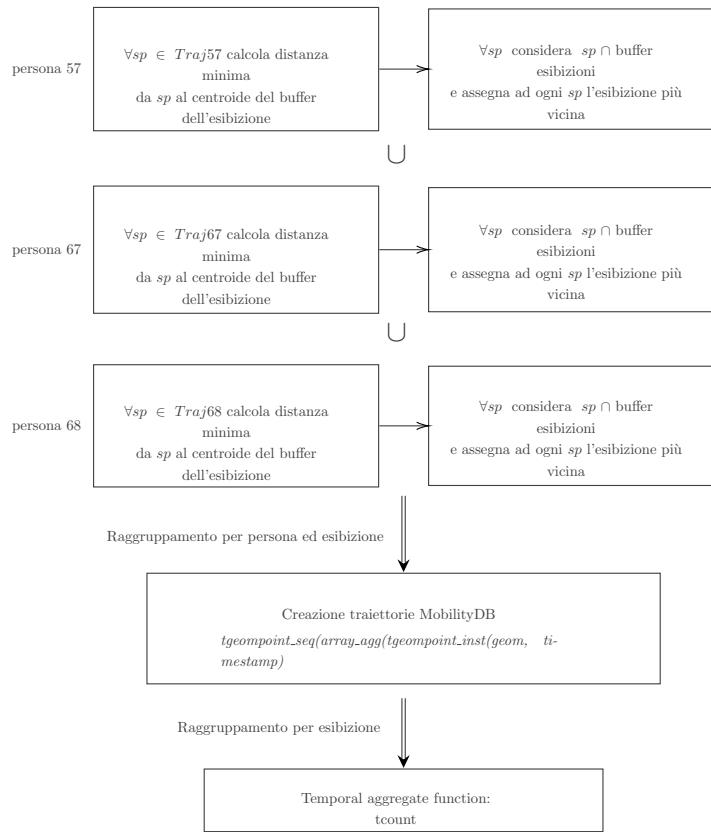


Figura 8: Diagramma a blocchi della query.

Riferimenti bibliografici

- [1] Ramaswamy Hariharan and Kentaro Toyama. Project lachesis: Parsing and modeling location histories. volume 3234, pages 106–124, 10 2004.
- [2] Tales Nogueira, Clayson Celes, Hervé Martin, Antonio Loureiro, and Rosana Andrade. A statistical method for detecting move, stop, and noise: A case study with bus trajectories. 9:214–228, 12 2018.
- [3] Tao Wu, Huiqing Shen, Jianxin Qin, and Longgang Xiang. Extracting stops from spatio-temporal trajectories within dynamic contextual features. *Sustainability*, 13(2), 2021.
- [4] Longgang Xiang, Meng Gao, and Tao Wu. Extracting stops from noisy trajectories: A sequence oriented clustering approach. *ISPRS International Journal of Geo-Information*, 5(3), 2016.
- [5] Yu Zheng. Trajectory data mining: An overview. *ACM Transaction on Intelligent Systems and Technology*, September 2015.
- [6] Esteban Zimányi, Mahmoud Sakr, and Arthur Lesuisse. MobilityDB: A mobility database based on PostgreSQL and PostGIS. *ACM Trans. Database Syst.*, 45(4), December 2020.