Progetto Gestione dell'Informazione Geospaziale

Flavio Forenza

Matricola: 938367

1 Introduzione

Nel seguente documento vengono descritte tutte le fasi riguardanti la gestione e la manipolazione dei dati che costituiscono due traiettorie. Queste ultime sono state ricavate in un ambiente "indoor", mediante il tracciamento di due dispositivi posseduti da due individui diversi. Lo studio quindi, è centrato nell'esaminare quali relazioni abbiano tali traiettorie, in particolare si vogliono ricercare alcune zone, comuni e non, attraversate da codeste. Per far ciò verranno analizzati i tratti in cui i due individui camminano assieme e/o si fermano. La struttura dell'elaborato è composta dalle seguenti fasi:

- ***Librerie e Tecnologie utilizzate***
- ***Analisi dei dati***
- ***Acquisizione dei dati***
- ***Operazioni di manipolazione dei dati***
    1. *Filtraggio dei dati*
    2. *StayPoints Detection (aree di stop in comune)*
    3. *Clustering K-Means*
    4. *Interpolazione*
    5. *Identificazione dei cammini*
    6. *Creazione e salvataggio traiettorie (.shp) in locale*
    7. *Salvataggio dati in PostgreSQL*

2 Librerie e Tecnologie utilizzate

Prima di poter proseguire con la lettura dell'elaborato, è fondamentale disporre di un ambiente di sviluppo che integrasse Python come linguaggio di programmazione. In questo caso, è stato utilizzato Python v.3.7.4, mentre l'ambiente di sviluppo è Spyder (Anaconda). Le librerie necessarie alla realizzazione di alcuni concetti, che richiedono pertanto la loro installazione e l'importazione all'interno dell'*IDE* sono:

- *Pandas*: gestione ed analisi dei dati;
- *Geopandas*: gestione dati geospaziali;
- *Shapely*: gestione delle geometrie;
- *Sklearn*: cluster K-Means;
- *Sqlalchemy*: ORM per la connessione ad un database
- *Sqlalchemy-utils*: gestione database
- *Fiona*: Gestione file in fromato .shp (shapefile)

Dopo queste, seguono l'implementazione diverse librerie di sistema come: *Statistics, Math, Numpy, Collections, Os, Shutil.*

3 Analisi dei dati

Il trattamento dei dati viene effettuato su due file, entrambi aventi formato shapefile (.shp), nella quale sono presenti diversi attributi. Le informazioni più importanti, riportate in questi ultimi, riguardano:

- *Time*: tempo di misurazione, con un intervallo pari a 20 decimi di secondo da ogni rilevazione;
- *x,y,z*: Coordinate cartesiane di ogni rilevazione. L'elemento z verrà trascurato in quanto le misurazioni vengono effettuate sullo stesso piano.
- *Timestamp*: Informazioni temporali composte da data e tempo di ogni rilevazione;
- *geometry*: Ogni misurazione viene rappresentata da una figura, ovvero un punto, e quindi una geometria.

Esaminando il numero di rilevazioni, ogni secondo vengono acquisite all'incirca cinque misurazioni da ogni singolo individuo. Come riportato nella traccia assegnata, alcuni dati, nel complesso, vanno a formare il "rumore" di rilevazione, pertanto verranno sottoposti a un'apposita fase di filtraggio.

4 Acquisizione dei dati

I due file forniti, aventi formato .shp, per essere manipolati nel linguaggio Python, sono stati importanti utilizzando una libreria già esistente allo stato dell'arte. Tale libreria prende il nome di "***GeoPandas***" (url: `http://geopandas.org/`). Geopandas è una libreria open source utile alla manipolazione dei dati Geospaziali in Python. La sua funzione è quella di estendere i tipi di dati utilizzati dalla famosa libreria "***Pandas***" (url: `https://pandas.pydata.org/`), per consentire operazioni spaziali su tipi geometrici. In particolare, pandas

mette a disposizione una struttura dati tabulare bidimensionale, avente righe e colonne etichettate ed una dimensione variabile. Un oggetto (dizionario, lista etc.) avente delle coordinate, viene convertito in Python utilizzando il Data Frame nella quale, grazie al Geo Data Frame, messo a disposizione da GeoPandas, gli sarà assegnato un sistema di coordinate (CRS) e una colonna "*geometry*" raffigurante la geometria di ogni coordinata. Le operazioni geometriche, che Geopandas mette a disposizione, sono eseguite dal tool "***Shapely***" (url: `https://pypi.org/project/Shapely/`), il quale permette la manipolazione l'analisi di oggetti geometrici planeari.

5 Operazioni di manipolazione dei dati

Dopo aver esaminato la struttura dei dati, si possono applicare diverse operazioni utili a poter condurre lo studio. La prima operazione importante da eseguire, sulla quale si basano altre operazioni, è il filtraggio dei dati.

5.1 Filtraggio dei dati

La seguente operazione è utile per poter rimuovere, o quantomeno diminuire, la presenza del rumore. Ritornando all'analisi dei dati, si può notare che la prima traiettoria *oneway-meetingA* ha un tempo (timestamp) iniziale pari a 12:02:17, fino ad arrivare ad un tempo finale di 12:05:58, per un totale di 3:41 minuti trascorsi. La seconda traiettoria, *oneway-meetingB*, invece ha un tempo pari a 12:02:09, mentre quello finale corrisponde al tempo finale della prima traiettoria, ovvero 12:05:58, per un totale di 3:49 minuti trascorsi. Facendo una conversione in secondi, la prima traiettoria viene composta in:

$$(3x60) + 41 = 221s$$

Mentre la seconda traiettoria viene costruita utilizzando un tempo, in secondi, pari a:

$$(3x60) + 49 = 229s$$

Da come si può notare, la costruzione della prima traiettoria viene fatta utilizzando un tempo minore rispetto alla seconda traiettoria. Ricordando che in un secondo vengono prese all'incirca cinque misurazioni (ogni venti decimi di secondo), teoricamente la seconda traiettoria dovrebbe avere più rilevazioni rispetto alla prima. Visionando i dati, accade proprio il contrario, ovvero la prima traiettoria è composta da un numero di punti (1094) maggiore rispetto al numero di punti (1048) che compongono la seconda traiettoria. Questo fenomeno mostra come vi sia la presenza di rumore legato alle modalità di acquisizione della posizione, magari influenzato proprio dalla presenza di ostacoli nell'ambiente o dalla strumentazione utilizzata. La questione del rumore non è presente solamente in questi fattori, ma anche nella posizione rilevata, infatti, anche visivamente, si può notare la presenza di alcuni punti (outliers) che si separano dalle traiettorie. Di seguito, sono riportate le traiettorie originali:

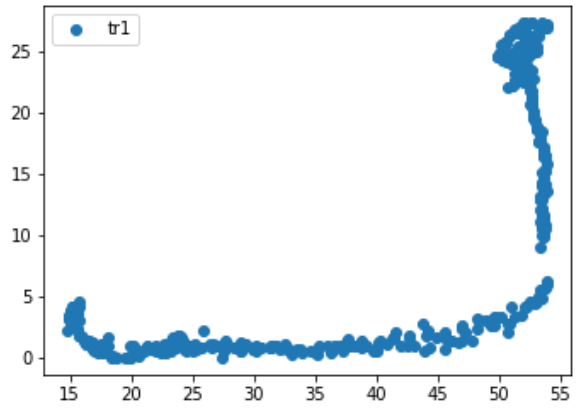

Figure 1: Traiettoria 1

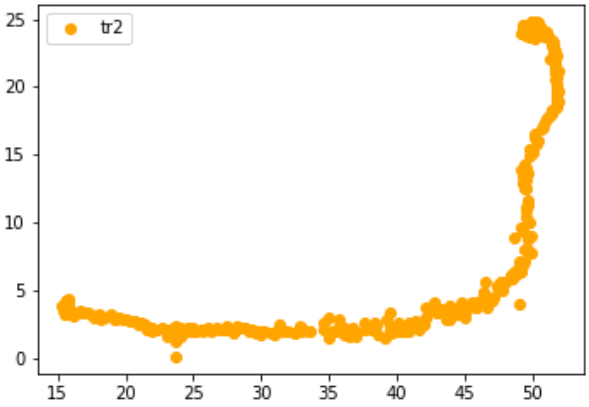

Figure 2: Traiettoria 2

Mentre la seguente figura mostra entrambe le traiettorie sullo stesso piano:

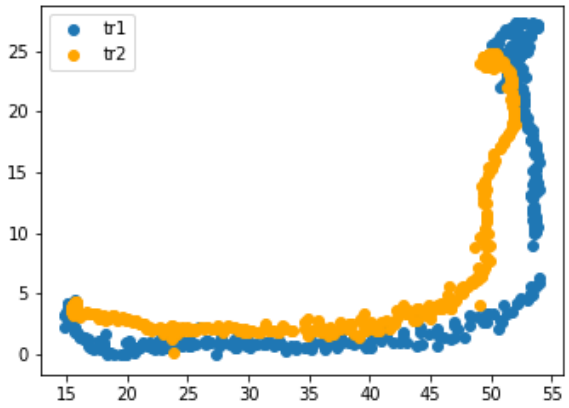

Figure 3: Traiettorie sullo stesso piano

Per realizzare il filtraggio, l'idea è quella di considerare tutte le misurazioni, effettuate in ogni secondo (5 di solito), e di ricavarne una singola posizione sfruttando il calcolo della ***mediana*** fra di esse. In questo modo si otterranno delle misurazioni che non tengono conto degli outliers, ma della maggior parte dei punti che risultano essere validi. Quest'operazione in altre applicazioni, come ad esempio nell'Image Processing, viene svolta di solito dal cosiddetto "*Filtro Mediano*" il quale, data una serie di punti, ne calcola solo uno avente come coordinata x la mediana di tutte le x che costituiscono i punti, e come coordinata y la mediana di tutte le y appartenenti agli stessi punti. L'elemento mediano rappresenta il cinquantesimo percentile di un insieme ordinato di numeri. Applicando tale filtro, il numero di misurazioni sarà uguale al numero di secondi trascorsi, e quindi per ogni traiettoria avremmo:

- Traiettoria 1: 221 misurazioni
- Traiettoria 2: 229 misurazioni

Riducendo il numero di punti, le traiettorie risultanti sono le seguenti:

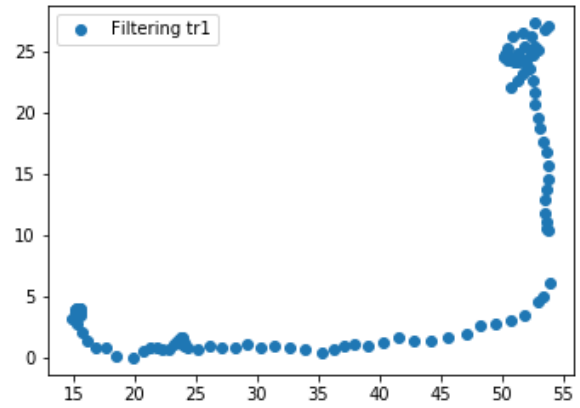

Figure 4: Traiettoria 1 filtrata

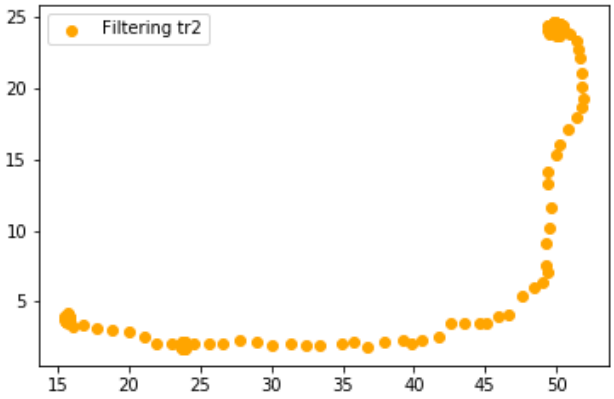

Figure 5: Traiettoria 2 filtrata

Mentre la seguente figura mostra entrambe le traiettorie filtrate sullo stesso piano:

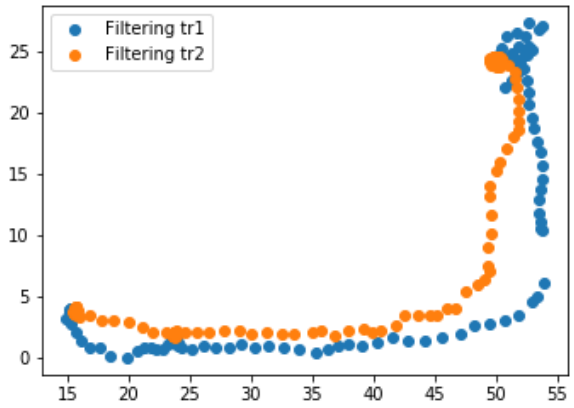

Figure 6: Traiettorie filtrate sullo stesso piano

Come si evince nelle figure precedenti, le traiettorie sono visivamente più intuitive e pulite rispetto alle originali, andando a ridurre il numero di outliers presenti in queste ultime. Per ottimizzare le operazioni successive, sono stati rimossi i primi 8 punti della seconda coordinata in quanto questa ha un numero di secondi maggiore rispetto a quello della prima traiettoria, quindi entrambe le traiettorie da ora in poi avranno 221 punti, equivalente al numero di secondi.

5.2 StayPoints Detection

In questo sotto-paragrafo viene discussa un'operazione utile a individuare le posizioni (*StayPoints*) in cui gli individui sostano, o si fermano per un certo periodo di tempo. Quando un individuo permane in un'area per molto tempo, a livello geometrico produrrà una sovrapposizione di punti che stanno ad indicare la sua posizione in quell'intervallo di tempo. Pertanto, si verranno a creare delle aree che contengono una quantità di punti maggiore rispetto ad altre. Il problema sta nell'isolare tali aree per poter individuare le zone comuni dove gli individui si fermano. L'idea è quella di calcolare la distanza di ogni punto dal suo successivo utilizzando la *Distanza Euclidea* e, in seguito, di scartare tutti quei punti le cui distanze risultano essere maggiori rispetto al valore mediano di tutte distanze dei punti appartenenti ad una singola traiettoria. Quindi, una volta aver ricavato la lista delle distanze di ogni punto con il suo successivo, si andrà a calcolare il valore mediano di tutte le distanze.Tale valore mediano viene preso direttamente dai seguenti boxplot, ognuno appartenente ad ogni traiettoria:

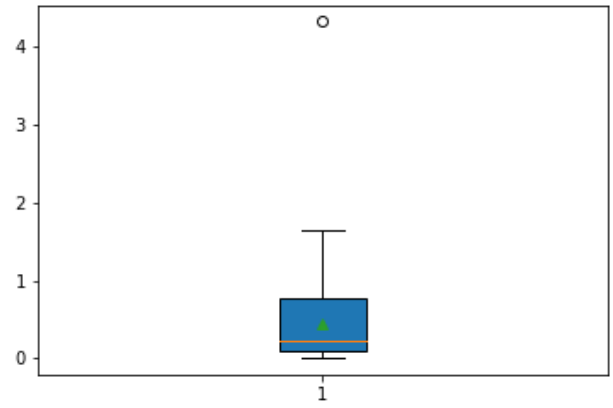

Figure 7: Box plot distanze punti Traiettoria 1

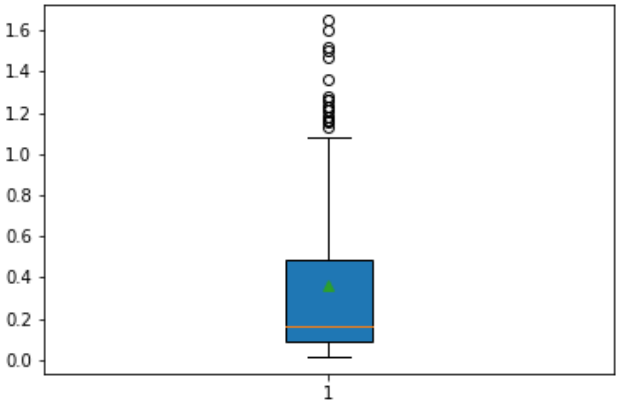

Figure 8: Box plot distanze punti Traiettoria 2

E' chiaro che il confronto di una distanza, con un valore mediano di tutte le distanze, porterà ad una riduzione dei punti presenti sulla traiettoria, facendo prevalere quelle zone in cui vi è una concentrazione maggiore di punti (dove si ferma il singolo individuo) a discapito delle zone in cui vi sono pochi punti (cammino che porta da un'area di stop all'altra). Quindi la presenza di zone in cui vi sono un alto numero di punti, va a influenzare il calcolo della mediana. Il

confronto di ogni singola distanza fra due punti, e la mediana di tutti i punti, ci porta a filtrare un numero elevato di punti, fino ad ottenere il seguente risultato:

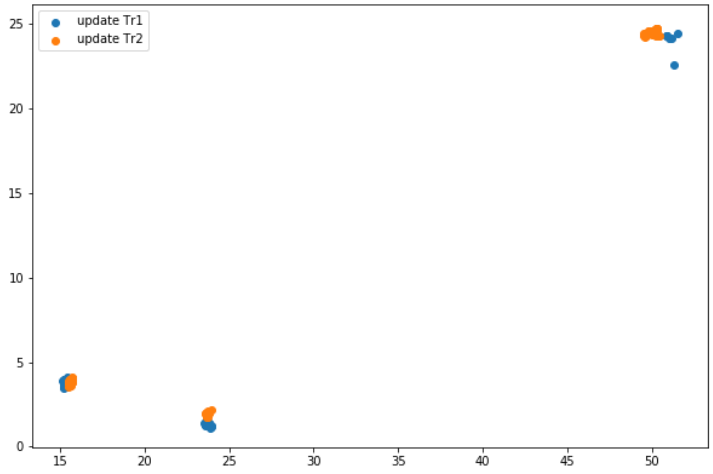

Figure 9: Aree di stop in comune (StayPoints)

L'immagine mostra le zone di stop in comune fra i due individui. Ricordo che in tali aree gli individui hanno trascorso un tempo maggiore rispetto al tempo di percorrenza utile per passare da un'area all'altra. Ovviamente i punti risultanti, di ogni traiettoria, sono racchiusi in un intorno di punti, pertanto vanno considerati tutti quei punti che non si discostano di molto da codesti. Facendo attenzione al risultato prodotto, possiamo eliminare i singoli punti che provocano rumore, come quello in alto a sinistra appartenente alla prima traiettoria, utilizzando un meccanismo più sofisticato che prende il nome di clustering.

5.3 Clustering K-Means

Riprendendo i punti ricavati nel paragrafo precedente, possiamo applicarci un metodo di clusterizzazione basato sul centro (***center-based***), in modo da manipolarli a nostro piacimento. Per esempio, può tornare utile effettuare una separazione di ogni gruppo, in modo da conteggiare i singoli punti presenti in essi e prendere delle decisioni in merito. L'idea alla base è quella di creare un'unica geometria, per ogni gruppo, che comprendesse tutti i punti al loro interno. Si vuole creare quindi un *buffer*, di ogni cluster, che comprendesse diversi punti situati sempre vicino ai gruppi precedentemente rilevati, in modo da approssimare la zona in cui un individuo ha sostato. Per poter creare un buffer, ogni gruppo dovrebbe disporre di un *centroide*, il quale può essere calcolato direttamente dall'algoritmo di clusterizzazione ***K-Means***. Il K-Means è un algoritmo di *apprendimento non supervisionato* che trova un numero fisso di

cluster in un insieme di dati. Ogni cluster è composto da punti, o oggetti, che hanno una certa somiglianza fra loro, che vengono chiamati ***data points***. Per poter applicare il K-Means bisogna, per ogni cluster, definire un centroide, ossia un punto al centro del cluster, formato grazie al calcolo della media delle coordinate appartenenti ai punti che compongono tale cluster. L'algoritmo analizza ciascuno dei data points e li assegna al centroide più vicino. Quindi viene calcolata la *distanza Euclidea* tra ogni data points e ogni centroide. In termini matematici si ha che:

$$\arg\min_{c_j \in C} dist(c_j, x)^2$$

dove c_j è un centroide nell'insieme C, ovvero l'insieme che include tutti i centroidi, mentre le x sono i data points e $dist()$ è la distanza euclidea standard. Di seguito, viene rappresentata un'immagine avente al suo interno tutti i centroidi che rappresentano le aree in cui ogni individuo si ferma:

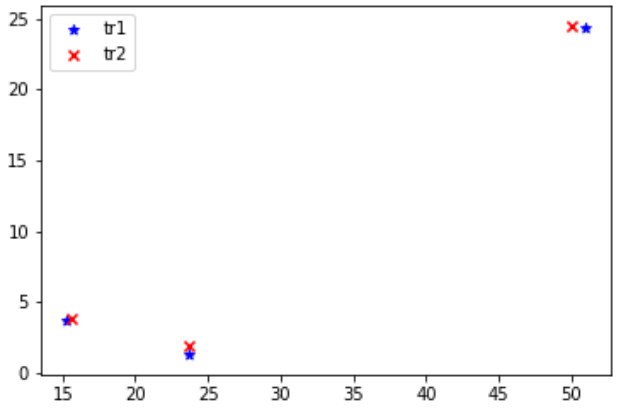

Figure 10: Centroidi appartenenti ad ogni cluster

In maniera tabellare, i centroidi di ogni cluster sono i seguenti:

	Traiettoria 1		**Traiettoria 2**	
	x	**y**	**x**	**y**
Cluster 1	23.7198	1.33712	23.7236	1.96597
Cluster 2	50.9171	24.3319	50.0375	24.4358
Cluster 3	15.3206	3.78145	15.6395	3.87138

Table 1: Centroidi di ogni cluster

Per dare una visualizzazione globale del numero di punti che compongono ogni cluster, ogni centroide viene prima trasformato in un punto geometrico e,

successivamente, si utilizza codesto per costruire un *Buffer*, ognuno di egual misura, utile a vedere quanti punti sono racchiusi in ogni area di stop:

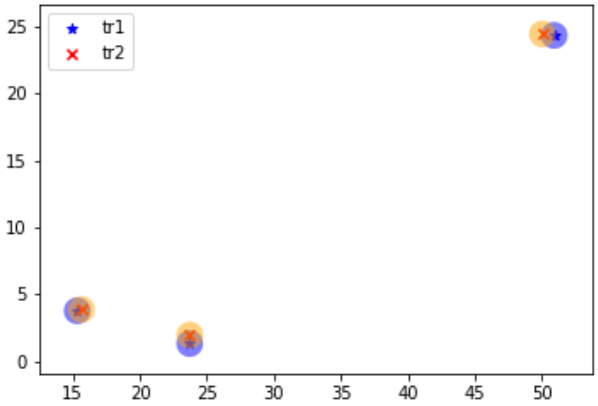

Figure 11: Buffer di ogni centroide

Confrontando con le traiettorie, ogni buffer racchiude le seguenti zone di stop:

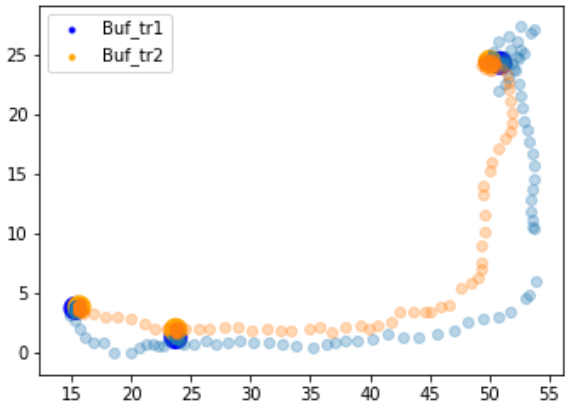

Figure 12: Buffer e Traiettorie

Dopo un'analisi del comportamento dell'algoritmo, per ogni traiettoria, il K-Means individua, in maniera sequenziale, i seguenti cluster:

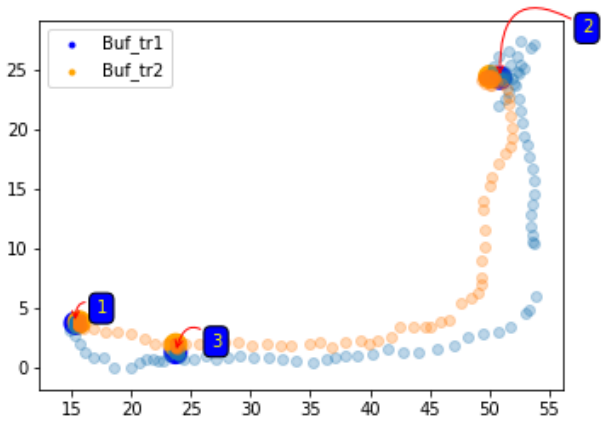

Figure 13: Sequenza di identificazione cluster K-Means

E' risultato fondamentale conteggiare automaticamente il numero di punti contenuti in ogni cluster. Nella seguente tabella, vengono riportati i numeri dei punti appartenente ad ogni cluster:

	Traiettoria 1	**Traiettoria 2**
Cluster 1	79	70
Cluster 2	7	19
Cluster 3	25	23
Totale	**111**	**112**

Table 2: Punti in ogni cluster

Come possiamo vedere dalla tabella, i numeri presenti in ogni cluster non differiscono di molto. Prendendo in considerazione il ragionamento effettuato in fase di filtraggio del rumore, ogni punto, in questa tabella, sta a specificare ogni secondo trascorso in ogni area. Nello specifico, guardando il risultato posto nella riga del totale, gli individui differiscono di 1 solo secondo fra loro. Questo vuol dire che più o meno i cluster contengono lo stesso numero di punti e che entrambi gli individui hanno trascorso lo stesso tempo nelle aree di stop. Si può quindi dedurre che entrambi gli individui hanno impiegato un tempo pari circa alla metà del tempo totale (221s) nelle aree individuate dai clusters, mentre hanno impiegano la restante parte del tempo nei percorsi che uniscono le tre aree. Dal ragionamento appena effettuato, si evince che entrambi gli individui si spostano insieme, passando da un'area di stop all'altra. In aggiunta, se volessimo visualizzare le aree che intersecano le aree comuni di entrambe le traiettorie, di seguito troviamo un'***intersezione*** eseguita fra i buffer di ogni centroide:

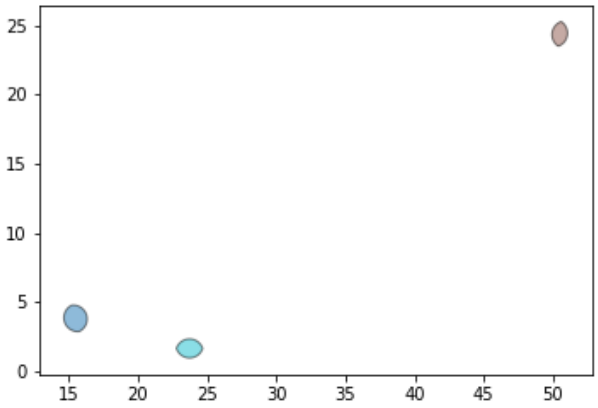

Figure 14: Operazione di Intersezione dei Buffers

Per valutare la bontà dei cluster ottenuti, possiamo calcolarci l'indice **SSE (Sum of Squared Error)** ovvero la *Somma del Quadrato degli Errori*, utilizzando la seguente formula:

$$SSE = \sum_{i=1}^{k} \sum_{x \in C_i} d^2(m_i, x)$$

dove k rappresenta il numero dei cluster identificati, x è un punto appartenente al cluster C_i, mentre m_i è il centroide di quel cluster ed infine d^2 è la distanza, al quadrato, che hanno il punto e il centroide. La tabella seguente mostra, per ogni cluster appartenente ad ogni traiettoria, la corrispettiva somma delle distanze al quadrato, con l'indice SSE che rappresenta la somma derivante dalla somma delle distanze al quadrato di ogni cluster:

	Traiettoria 1	**Traiettoria 2**
Cluster 1	0.743324	0.0870391
Cluster 2	0.0700323	0.084633
Cluster 3	0.0053518	0.10445
SSE	0.818708	0.276122

Table 3: SSE Clusters

Come si può notare, solamente il primo cluster, appartenente alla prima traiettoria, ha una sommatoria delle distanze al quadrato più alta rispetto alle altre. Questo, rispetto agli altri cluster, è causato dal fatto che i punti appartenenti a tale cluster sono più sparsi e più distanti dal proprio centroide. Isolando

solamente questo caso, si considera positivamente il lavoro effettuato dal K-Means. Sicuramente, si avranno riscontri positivi nell'utilizzare un approccio di validazione del cluster basato sulla **validazione interna**, la quale sfrutta le proprietà degli **indici interni**. Questi, basati sulla nozione di *compattezza* e *separazione*, confermerebbero la correttezza del cluster creato, ovvero che il cluster rispetta il principio di compattezza, che si verifica quando i punti in un cluster sono vicini tra loro (bassa distanza intra-cluster fra punti), ed inoltre rispetta anche il principio di separazione, cioè quando i punti di cluster diversi si trovano ad una distanza tale da separarli (alta distanza inter-cluster). Si fa notare che, la scelta di tale algoritmo di partizionamento è stata effettuata proprio perchè sono avvenute due operazioni importanti, ognuna delle quali ha permesso un'ottima operabilità di tale algoritmo; Tali operazioni sono:

- ***Operazione di Pre-processing***: questa operazione, già effettuata in precedenza, si basa nella rimozione degli outliers in entrambe le traiettorie di partenza: Questa operazione ha permesso quindi di fornire in input, all'algoritmo K-Means, direttamente le traiettorie filtrate;
- ***Operazione di Post-processing***: tale operazione è quella effettuata nel momento in cui sono stati rimossi quei cluster di piccole dimensioni, che erano costituiti da pochi punti.

5.4 Interpolazione

A volte i dati, che compongono i file, risultano essere mancanti o addirittura danneggiati, provocando, come in questo caso, l'assenza di alcune sequenze di punti che compongono alcuni tratti di ogni traiettoria. L'idea è quella di ricostruire tali parti mancanti utilizzando una tecnica detta "***Interpolazione***". L'operazione di interpolazione ha il compito di individuare nuovi punti, del piano cartesiano, a partire da un insieme finito di punti già noti. In Python, la una libreria che mette a disposizione tale tipo di operazione prende il nome di *Shapely*, precedentemente citata nel paragrafo 3. Questa utilizza la distanza di due punti, con lo scopo di inserire, fra codesti, n punti mancanti, andando a fare la media fra le coordinate dei due punti e quelli appena creati, il tutto in maniera iterativa. Il procedimento utilizzato è quello effettuato da una comune "***Interpolazione lineare***", la quale ha lo scopo di trovare approssimativamente un valore compreso tra due valori noti. Esempio, immaginiamo di avere due punti, i quali rappresentano gli estremi, e decidiamo di voler trovare una sequenza di n punti distinti fra i due. L'algoritmo calcolerà prima la distanza fra i due estremi e genererà il nuovo punto che sarà interposto al centro. In maniera iterativa, per il calcolo di un nuovo punto, verrà effettuato lo stesso calcolo prendendo come punto estremo il punto creato precedentemente, e così via. E' stata creata una condizione che l'algoritmo deve rispettare, ovvero l'algoritmo aggiunge un punto nuovo, alla sequenza di punti da calcolare, ogni qualvolta i punti non rispettano una distanza massima equivalente alla distanza media ottenuta dai punti della prima traiettoria filtrata. Riferendoci sempre al caso di studio, tale operazione viene effettuata quindi sulla prima traiettoria, nella

quale sembra mancare una porzione di punti. Ovviamente, è possibile applicare il concetto di interpolazione anche alla seconda traiettoria in base alle proprie esigenze. Si può constatare, tramite il box plot in figura 5.2 ,rappresentante le distanze fra un punto e il suo successivo, che sussiste una distanza fra due punti maggiore di 4, lungo l'asse delle ordinate. L'immagine seguente mostra la collocazione (ellisse in rosso) di tale porzione avente tale distanza fra i due punti:

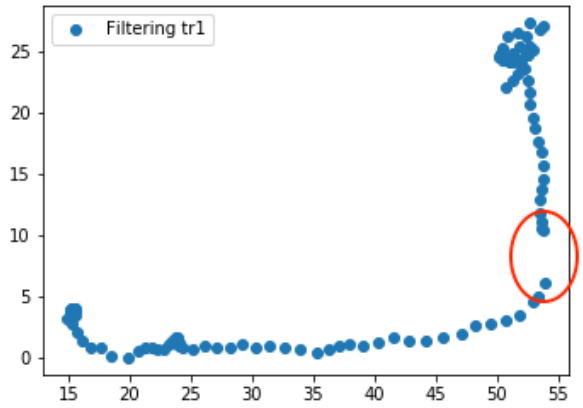

Figure 15: Sezione traiettoria mancante

Applicando l'algoritmo precedente, che sfrutta la tecnica dell'interpolazione, questo è riuscito a trovare ulteriori quattro punti appartenenti al segmento mancante, ottenendo il seguente risultato:

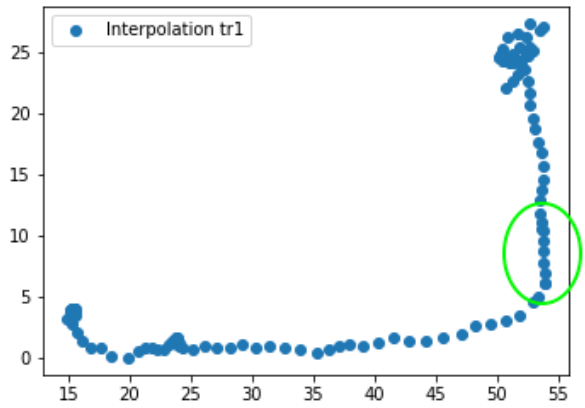

Figure 16: Interpolazione punti mancanti

I risultato dell'operazione di interpolazione risulta essere più veritiero con segmenti mancanti di piccole dimensioni, mentre la ricostruzione di segmenti, di dimensioni maggiori, potrebbe aggiungere punti mai percorsi da chi sta compiendo tale traiettoria. Questa operazione è quindi utile per ricostruire quelle posizioni in cui non è stato possibile rilevare un movimento a causa di diversi fattori come ostacoli, barriere o malfunzionamento temporaneo del dispositivo di acquisizione.

5.5 Identificazione dei cammini

Quando si hanno a disposizione due o più traiettorie, oltre a identificare le aree occupate da entrambe, si può effettuare un'analisi sulle relazioni dei loro cammini. In questo sotto paragrafo si andranno ad identificare tutti quei percorsi che hanno entrambe le traiettorie vicine, o distanti, con lo scopo di capire quando i due individui sono ad una distanza ravvicinata, o lontani, durante il loro cammino. Per effettuare tale studio, la tecnica utilizzata prende il nome di ***Range Interquartile (IQR)***. In statistica, l'indice *IQR*, detto anche *Scarto Interquartile Campionario*, sta ad indicare la differenza tra il terzo e il primo quartile, ovvero l'ampiezza della fascia di valori che contiene la metà "Centrale" dei valori osservati. Ricordo che i quartili sono quei valori che ripartiscono la popolazione in quattro parti di uguale numerosità. Il primo quartile *Q1* è definito come il numero medio tra il numero più piccolo e la mediana, il secondo quartile *Q2* rappresenta la mediana ed infine il terzo quartile *Q3* rappresenta il numero medio tra la mediana e il valore più grande dell'insieme di dati. Lo scarto interquartile campionario rappresenta una misura di quanto i valori si allontanino dalla media campionaria. Il suo calcolo è dato dalla differenza tra il terzo ed il primo quartile:

$$IQR = Q3 - Q1$$

Nel nostro caso, l'IQR serve a prendere tutti gli outliers di una distribuzione contenente tutte le distanze fra due punti, con lo stesso timestamp. In particolare, tale indice va a identificare i punti che hanno una distanza che si discosta dagli estremi del primo e del terzo quartile, ovvero quei punti che hanno una distanza notevole rispetto ai punti, racchiusi ne quartili 1 e 3, a piccola distanza. Affinché tale identificazione vada a buon fine, si deve moltiplicare, all'IQR, il valore *1.5*, sommare il risultato al valore del quartile, ed infine confrontare il tutto con la distanza:

$$d_i > Q3 + 1.5 * IQR \Leftrightarrow d_i < Q1 + 1.5 * IQR$$

dove d_i rappresenta la distanza fra i due punti. In maniera visiva, gli outliers presi in considerazione sono tutti quelli rappresentanti nel seguente box plot:

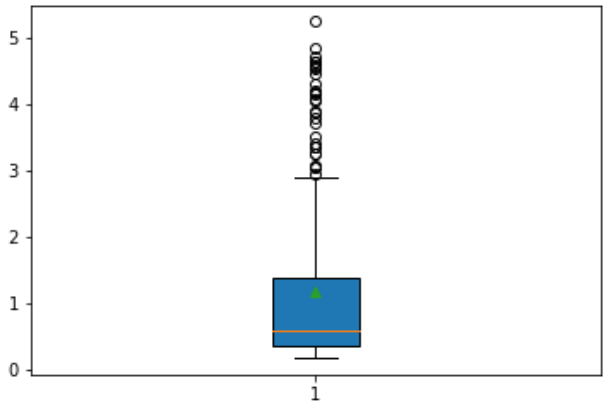

Figure 17: Box Plot outliers

In maniera più precisa, gli outliers si collocano tutti dopo il terzo quartile, ad una distanza maggiore o uguale a 2.91, fino ad arrivare a 5.23. Per poter definire i *Cammini Distanti* delle due traiettorie, si prendono tutti gli outliers che sono compresi fra i due valori precedenti e, andandoli a proiettare sul piano cartesiano, otteniamo i seguenti cammini:

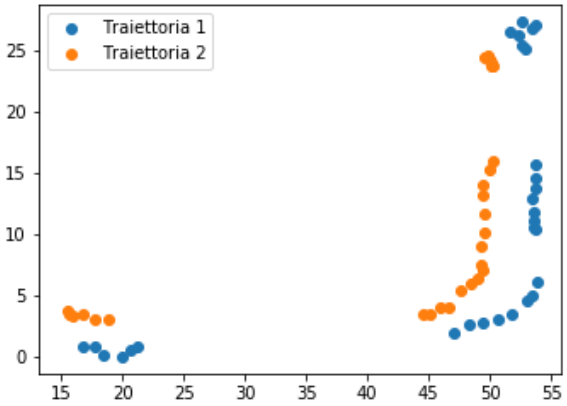

Figure 18: Cammini distanti

E' stata restituita questa tipologia di cammino, in quanto gli individui hanno deciso di separarsi, magari attraversando porte diverse o a causa di ostacoli,

come oggetti o persone, che erano di intralcio. Per determinare i *Cammini Vicini*, possiamo prendere i restanti punti che non appartengono ai cammini precedenti:

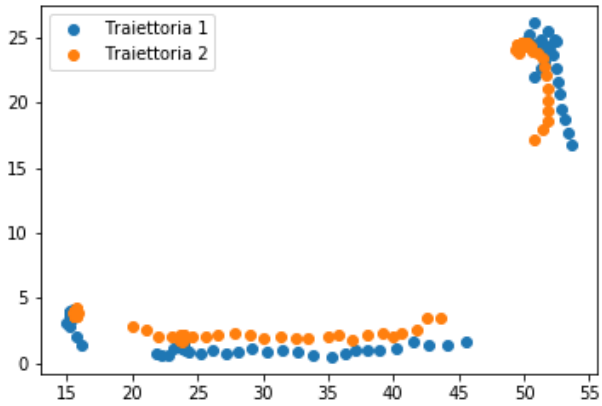

Figure 19: Cammini vicini

In questo modo quindi, sono stati identificati due tipologie di cammini, a distanza variabile, appartenenti ad entrambi i soggetti.

5.6 Creazione e salvataggio Traiettorie (.shp) in locale

Per consentire la visualizzazione delle traiettorie create, si è deciso di salvarle tutte in formato *Shapefile* (.shp), in modo da porterle visualizzare in ambiente ***QGIS***. In particolare, gli elementi salvati in tale formato, riguardano:

1. Triettoria 1 filtrata (senza rumore);
2. Triettoria 2 filtrata (senza rumore);
3. Clusters punti Traiettoria 1 individuati dal K-Means;
4. Clusters punti Traiettoria 2 individuati dal K-Means;
5. Buffers dei cluster della Traiettoria 1 creati a partire dai centroidi di questi ultimi;
6. Buffers dei cluster della Traiettoria 2 creati a partire dai centroidi di questi ultimi;
7. Traiettoria 1 interpolata (con l'aggiunta di nuovi punti sulla traiettoria filtrata);

8. Cammino "Distante" Traiettoria 1;
9. Cammino "Distante" Traiettoria 2;
10. Cammino "Vicino" Traiettoria 1;
11. Cammino "Vicino" Traiettoria 2.

Ogni file memorizzato ha associato, come sistema di riferimento *(CRS)*, l'**EPSG 3003**, corrispondente al ***Monte Mario / Italy Zone 1 - Proiettato***, con unità espressa in *metri*. La memorizzazione dei file avviene direttamente nella directory di lavoro, dove è presente il progetto, dove verranno create delle sotto directory che conterranno le coppie di traiettorie, ognuna per ogni argomento discusso. Verranno create le seguenti cartelle:

1. Tr-Filter: contenente le traiettorie filtrate, prive di rumore;
2. K-Means: contenente i clusters di entrambe le traiettorie;
3. Buffer: conentente i Buffers di ogni geometria;
4. Interpolazione: contenente l'interpolazione della Traiettoria 1;
5. Cammini: contenente i cammini Distanti e Vicini di entrambe le traiettorie.

Per la creazione e la memorizzazione delle geometrie, sono stati utilizzati diversi tool già presenti allo stato dell'arte. Fra questi, oltre al già citato tool "" *Shapely*, è stato utilizzata la libreria ***Fiona*** (url: `https://pypi.org/project/Fiona/`) utile per la lettura, scrittura e assegnamento del sistema di riferimento, di ogni geometria.

5.7 Salvataggio dati in PostgreSQL

Oltre che a salvare i dati in locale, si è pensato di salvarli anche nel noto DBMS Relazionale ad Oggetti ***PostgreSQL***. Quest'operazione consente ai sistemi GIS di accedere facilmente a tale database, avendo in esecuzione un Server locale lanciato da PostgreSQL. Quest'operazione è composta da una prima fase che riguarda la connessione al Server per poter accedere al database. L'utility utilizzata, in grado di stabilire una connessione, è "***Sqlalchemy***" (url: `https://docs.sqlalchemy.org/en/13/`). Oltre a fornire la possibilità di connettersi, questo tool è in grado di effettuare query per interrogare la base di dati ed inoltre consente di aggiungere al database l'estensione "***postgis***" utile per poter inserire una colonna avente al suo interno delle geometrie con la possibilità di poterle gestire ed interrogare. Per poter inserire le geometrie, come i punti o i poligoni, ognuna di esse viene prima convertita in formato *WKT* alla quale viene associato il sistema di riferimento spaziale *SRID* specificato. Nel nostro caso, è stato impostato come *SRID* il *3003 Monte Mario*. Le credenziali utilizzate sono quelle di default, mentre dobbiamo specificare il nome del database che vogliamo creare:

- User: *postgres*
- Password: *password*
- IP: *127.0.0.1*
- Porta: *5432*
- Database: *Progetto Forenza*

I dati che si sono rilevati più interessanti da memorizzare sono gli stessi del paragrafo precedente, ovvero le tabelle memorizzate sono le seguenti:

- Tabella 1: Traiettoria 1 filtrata (senza rumore);
- Tabella 2: Traiettoria 2 filtrata (senza rumore);
- Tabella 3: Clusters punti Traiettoria 1 individuati dal K-Means;
- Tabella 4: Clusters punti Traiettoria 2 individuati dal K-Means;
- Tabella 5: Buffers dei cluster della Traiettoria 1 creati a partire dai centroidi di questi ultimi;
- Tabella 6: Buffers dei cluster della Traiettoria 2 creati a partire dai centroidi di questi ultimi;
- Tabella 7: Traiettoria 1 interpolata (con l'aggiunta di nuovi punti sulla traiettoria filtrata);
- Tabella 8: Cammino "Distante" Traiettoria 1;
- Tabella 9: Cammino "Distante" Traiettoria 2;
- Tabella 10: Cammino "Vicino" Traiettoria 1;
- Tabella 11: Cammino "Vicino" Traiettoria 2.

E' possibile verificare la correttezza di tale tabelle importando il database direttamente in *QGIS*, utilizzando il DB Manager, stabilendo una nuova connessione tramite ***PostGIS***, andando a riportare solamente il nome del database appena creato.