Alma Mater Studiorum · Università di Bologna

SCUOLA DI SCIENZE

Corso di Laurea in Informatica per il Management

Analisi ed identificazione di percorsi stradali tramite sensori inerziali

Relatore:
Dott.
Luca Bedogni

Presentata da:
Andrea Ercolessi

II Sessione
Anno Accademico 2018/2019

Indice

Introduzione **3**

1 Lo stato dell'arte **5**

1.1 La localizzazione: l'equilibrio tra servizio e rispetto della privacy 6

1.2 Localizzazioni DRS (Dead-Reckoning Systems) 7

1.3 Ambiti di applicazione 8

1.4 Studi correlati 9

2 Motivazioni **13**

3 Sviluppo **16**

3.1 Creazione del grafo 16

3.1.1 Architettura del sistema 16

3.1.2 NetworkX 18

3.1.3 OSMnx e la sua struttura 18

3.2 Simulazione dei percorsi 18

3.2.1 Estrazione lunghezza dei segmenti simulati 19

3.2.2 Estrazione delle angolazioni 21

3.3 Creazione dei sotto grafi 21

3.4 Unione dei nodi con la medesima angolazione 22

3.5 Algoritmo di ricerca dei sotto grafi 24

3.6 Raffinare i risultati di ricerca 27

3.7 Creazione dei file di Output 30

4 Analisi e risultati finali **31**
4.1 Implementazione dei grafici 31
4.2 Studio preliminare dei grafi 32
4.3 Studio della densità di probabilità 33
4.4 Analisi densità di probabilità delle città 34
4.4.1 PDF delle Città Europee 36
4.4.2 PDF delle Città Americane 43
4.5 Analisi dei percorsi con differenti rumori 48
4.5.1 Ansali grafica del rumore di tutte le città scelte 49
4.6 Analisi dei percorsi con differenti rumori di una città Europea 50
4.7 Analisi dei percorsi con differenti rumori di una città Americana 51

5 Conclusioni e sviluppi futuri **53**

Introduzione

Negli ultimi 2 decenni si è assistito ad una diffusione di massa di dispositivi mobili tecnologici, in primis smartphones: i progressi delle tecnologie costruttive unitamente al progressivo abbassamento dei costi hanno fatto in modo che tali dispositivi diventassero di uso comune nella mani di milioni di utenti finali. Oggigiorno, anche il più semplice degli smartphone offre non solo varie tecnologie per la connettività (GSM, WiFi, LTE, Bluetooth) ma anche vari altri sensori atti a migliorare l'esperienza utente ed aumentare i servizi a sua disposizione.

Una delle tematiche più importanti associate ai dispositivi mobili è quella della loro geo-localizzazione; un intero mercato si è sviluppato intorno ai cosiddetti LBSs (Location Based Services – Servizi Basati sulla Localizzazione). Grazie all'evoluzione hardware/-software dei dispositivi mobili, centinaia di milioni di utenti sono ormai raggiungibili attraverso gli LBSs di nuova generazione ([5]).

In questo lavoro si presentano delle moderne tecniche di localizzazione che sfruttano la dotazione di sensori ormai comune alla maggior parte degli smartphones e si discutono i rischi relativi alla violazione della privacy degli utenti. Nella discussione si fa riferimento ai sistemi mobili basati su sistema operativo Android, ma lo studio è estendibile ad altri sistemi analoghi esistenti.

Più nel dettaglio nell'elaborato verrà presentata la sola simulazione del possibile scenario reale, che cercherà di riprodurre quanto più esattamente possibile una rilevazione dei dati attraverso il movimento di uno smartphone, facendo affidamento ad algoritmi implementati per la generazione di percorsi reali simulati e la successiva ricerca all'inter-

no delle mappe di alcune città prese in esame. Ogni città verrà presa in considerazione come un grafo non orientato dove gli incroci corrispondono ai nodi e le strade agli archi. Gli archi contengono diverse informazioni che una volta analizzate permetteranno di raggiungere l'obbiettivo prefissato.

Nella parte finale verrà effettuata un'analisi tramite l'aiuto di graci e rappresentazioni grafiche di disparate città, sia per quanto riguarda un contesto Americano sia per un contesto Europeo per effettuare diversi confronti e mettere in relazione ciò che le diversifica o le accomuna. Verrà discusso anche delle architetture delle diverse città e come queste inuenzano il successo o il fallimento dell'obbiettivo, facendo confronti su piante di città a scacchiera ed altre radiocentriche. Le piante radiocentriche sono caratterizzate dalla convergenza di svariate vie di comunicazione verso un punto centrale; tale modello si trova maggiormente nelle città con conformazione pianeggiante. Per quanto concerne la pianta a scacchiera, invece, si intende uno sviluppo delle strade della città secondo uno schema a griglia con isolati e forme geometriche bene denite; tale architettura è tipica di molte città americane.

Capitolo 1

Lo stato dell'arte

Oggi giorno gli smartphone offrono molteplici servizi, uno dei servizi fondamentali proposti è il GPS (Global Positioning System). L'uso di tale sensore viene impiegato per la localizzazione dei dispositivi. Gli smartphone sono però caratterizzati da numerosi sensori oltre il GPS come ad esempio giroscopio, magnetometro, accelerometro i quali, a differenza del GPS, non necessitano dei permessi da parte dell'utente; in tal modo un'applicazione potrebbe raccogliere i dati di questi sensori per analizzarli e riuscire a tenere traccia della posizione dell'utente durante la guida anche senza che esso se ne renda conto.

A tal proposito bisogna citare l'articolo riguardante uno studio condotto da un team di ricercatori del College of Computer and Information Science Northeastern University di Boston ([16]). Lo studio si basa su dati realmente raccolti tramite un'applicazione, creata dagli stessi autori, che registra i movimenti dello smartphone all'interno di un veicolo sfruttando proprio i sopra citati sensori, con lo scopo di dimostrare con quale relativa facilità è possibile violare la privacy di un utente.

Un altro studio effettuato invece per scopi del tutto decorosi è stato effettuato dal Università di Bologna ([10]) il quale vuole far emergere come i sistemi di navigazione possano essere basati su sensori inerziali raggiungendo lo stesso un ottima precisione dei dati del classico GPS.

1.1 La localizzazione: l'equilibrio tra servizio e rispetto della privacy

La localizzazione di un dispositivo mobile permette indubbiamente di facilitare molte attività attraverso la loro contestualizzazione geografica: ad esempio, un utente che stia cercando qualcosa potrebbe ottenere risultati molto più rilevanti se il sistema di ricerca potesse identificarne la posizione. Tuttavia, come è facile immaginare, le applicazioni che un utente può installare sul proprio smartphone non sempre hanno una provenienza certa ed uno sviluppatore malizioso potrebbe sfruttare i sensori del dispositivo per localizzarlo ad insaputa del suo proprietario e perseguire quindi scopi meno nobili rispetto a quello di offrire un utile servizio. Lo studio dei ricercatori di Boston ha per l'appunto evidenziato come sia possibile dedurre informazioni relative alla vita privata di una persona (ad esempio individuarne l'abitazione) basandosi semplicemente sulle rilevazioni di un sensore GPS.

Per questo motivo Android limita l'accesso al sensore GPS (che tra l'altro notoriamente gode di una buona precisione in ambienti esterni) da parte delle applicazioni. Il sistema operativo implementa un sistema di autorizzazioni secondo cui l'utente deve esplicitamente consentire alle applicazioni scaricate l'uso di determinate tipologie di sensori, in particolari quelli che possono indurre una violazione dei dati personali e della vita privata. Questa misura di sicurezza non è tuttavia sufficiente a garantire del tutto la privacy dell'utente perché, come il mondo della ricerca ha evidenziato ([3]), esistono tecniche di localizzazione alternative al GPS che possono funzionare anche ad insaputa dell'utente.

Un ramo in particolare, quello di maggiore interesse per questo lavoro, è quello della localizzazione a mezzo dei sensori inerziali:

- Accelerometro (rileva le componenti di accelerazione su 3 assi nello spazio)
- Giroscopio (rileva i cambiamenti nella velocità angolare)
- Magnetometro (consente di rilevare il nord magnetico del pianeta)

Allo stato attuale, questi sensori non sono considerati critici e un'applicazione software può leggere le misurazioni di questi sensori senza l'esplicito consenso dell'utente. In altri termini, un'apparentemente innocua applicazione (ed esempio una funzionalità torcia, si veda il caso americano [6]) può ingannare l'utente avviando un servizio in background che interroga periodicamente tali sensori ricavandone dati utili alla localizzazione. Anche ipotizzando che l'applicazione si serva di un server esterno di elaborazione dei dati, il volume di trasmissione (stimato essere pari a circa 80 KB per 1 ora di navigazione) sarebbe impercettibile da parte dell'utente nello scenario comune di trovarsi nella copertura di reti 3G o 4G (con una banda di upload che nel caso peggiore è 2Mbps, ma che può arrivare fino a 150 Mbps).

Negli esperimenti simulati è risultata una percentuale del 50% di individuare il percorso esatto compiuto dall'utente, mentre la percentuale scende al 30% per gli esperimenti di guida reale su strada. Chiaramente la percentuale di successo dipende molto dall'architettura stradale della città: una pianta stradale radiale contiene percorsi molto più facilmente identificabili rispetto ad una pianta a scacchiera, tipica di molte città americane.

1.2 Localizzazioni DRS (Dead-Reckoning Systems)

Le tecniche di localizzazioni DRS si basano su aggregazioni dati provenienti da sensori che permettono di ricostruire in maniera indiretta altre grandezze fisiche. Ad esempio, campionando la variazione di velocità di un corpo (accelerazione) con una certa frequenza temporale, è possibile calcolare la distanza percorsa; un discorso analogo si può fare campionando la variazione di velocità angolare per calcolare gli angoli di sterzata.

Uno degli studi che fornisce una base principale per la creazione di questo elaborato viene studiato nell'articolo presentato dall'università di Bologna descritto brevemente in precedenza. In questo studio è stata presentata una tecnica di localizzazione relativamente nuova e ancora molto esplorabile in termini di ricerca. Si tratta di una localizzazione DRS il quale sfrutta i dati di sensori inerziali per ottenere una serie di movimenti, gene-

ralmente derivati dalle variazioni di accelerazione, da cui è possibile ottenere il percorso e non utilizza il GPS (nemmeno allo scopo di acquisire una sorta di "punto geografico iniziale della navigazione"); al posto del GPS nella tecnica si sfruttano i dati pubblici di definizione delle mappe del mondo di OpenStreetMap ([15]) per dedurre, mediante un procedimento algoritmico, quale possa essere la posizione geografica dell'utente al termine di una sessione di guida. I sensori inerziali sfruttati sono l'accelerometro e il magnetometro (uno studio che utilizza unicamente il magnetometro è [16]). Ovviamente i risultati operativi degli esperimenti vanno accolti con la consapevolezza che i dati di OpenStreetMap (un progetto di pubblico dominio) hanno un'accuratezza variabile [17].

Il procedimento può essere schematizzato principalmente in tre punti:

- Identificazione dei segmenti, in cui identifichiamo il numero di segmenti stradali monitorando i cambiamenti dei dati del magnetometro (ad ogni sterzata significativa si identifica l'inizio di un nuovo segmento di viaggio);
- Algoritmo "Dead Reckoning System" (DRS), attraverso il quale calcoliamo la lunghezza di ogni segmento percorso grazie alle misure dell'accelerometro;
- Assegnazione dei segmenti ai dati OpenStreetmap, in cui si deduce il possibile percorso che è stato percorso dal dispositivo cercando sequenze di segmenti stradali simili nella distribuzione stradale reale scaricata da OpenStreetmap.

1.3 Ambiti di applicazione

I progressi della ricerca in questo settore possono avere una doppia valenza.

Dal punto di vista della protezione della privacy, questo studio può mettere in luce il livello di precisione a cui un potenziale malintenzionato può ambire in termini di localizzazione di un utente ignaro. La dimostrazione oggettiva dei risultati della ricerca può fare leva sugli sviluppatori di hardware e dei sistemi operativi verso l'implementazione di contromisure più adeguate a proteggere la localizzazione non autorizzata di un utente. Questo è un tema abbastanza sensibile, tanto che ci sono stati studi ([24]) concentrati

sulle modalità con cui offrire servizi basati sulla localizzazione (LBSs) con modalità tali da impedire, o comunque rendere difficile, l'identificazione di utenti specifici.

Dal punto di vista del consumo di energia, è opportuno evidenziare come la durata della batteria sia un fattore estremamente importante per un dispositivo mobile. Esistono già studi orientati a migliorare il rapporto tra precisione della localizzazione e consumo di energia dovuto ai sensori. L'implementazione di sistemi di localizzazione/navigazione di buona precisione basati su sensori inerziali anziché su sensori GPS costituisce un ulteriore passo in avanti nella riduzione del consumo energetico a parità di funzionalità.

1.4 Studi correlati

Al di là dei due studi analizzati in precedenza, le tecniche di localizzazione che non facciano uso dei sensori GPS sono state per molto tempo (e lo sono ancora) al centro dell'attenzione di molte ricerche al fine di superare alcuni tradizionali limiti del GPS stesso.

Uno di questi limiti è la precisione del GPS in ambienti interni, che non riesce a raggiungere lo stesso livello ottenibili in ambienti esterni. Si può passare da un errore effettivo di pochi metri (per l'ambiente esterno) e scenari molto peggiori, salvo i casi in cui non si riesca a ricevere per nulla il segnale del satellite a causa degli ostacoli costruttivi. Una perdita di precisione si può anche avere in scenari di navigazione che includono percorsi con molte gallerie e/o strade circondate di edifici molto alti che possono interferire con la ricezione del segnale. Non costituisce quindi una novità l'utilizzo di sensori inerziali per l'implementazione di sistemi di navigazione per ambienti interni (indoor navigation)[7].

Un studio introdotto dall'Università di Bologna ([8]) basato sull'uso dei sensori inerziali mostra come sia possibile capire la modalità di navigazione (TMD – Transportation Mode Detection), ovvero riesce rilevare il mezzo di trasporto adoperato dall'utente per effettuare uno spostamento senza l'aiuto del GPS. Viene fatto anche qui un riferimento

alla violazione della privacy in quanto un sistema in grado di individuare un percorso potrebbe usare queste informazioni per scopi non del tutto leciti. A seguito di quanto appena riportata sono stati fatti due studi ([11] e [12]) che cercano di analizzare il contesto in cui lo smartphone si trova e capire se l'utente utilizza un ascensore piuttosto che le scale per spostarsi tra i livelli di un edificio, ovvero cercano di estrapolare più informazioni possibili riguardo le abitudini quotidiane che un utente tende ad avere. I dati vengono poi elaborati ed usati in ambiti scientifico-medici.

Nello studio condotto dall'Università di Cambridge ([9]), gli autori hanno dimostrato che l'accelerometro e il giroscopio dello smartphone possono essere usati per identificare dei percorsi stradali indipendentemente dal verso in cui viene orientato il dispositivo e dalle condizioni di traffico presenti durante la misurazione.

La ricerca viene focalizzato su due punti fondamentali:

- Il cambiamento della velocità angolare rilevata dal giroscopio viene usato per identificare delle svolte durante la guida.
- Ed a ogni cambiamento viene generata una traccia unica composta di variazioni di velocità angolare.

In questo modo il dispositivo è in grado di riconoscere se l'utente ripete lo stesso percorso effettuato in precedenza basandosi sul confronto delle sequenze memorizzate. Per dimostrare la riuscita di tale studio sono state prese le velocità angolari da un telefono all'interno di un veicolo su un tipico tragitto che porta da un punto di inizio definito A sino un punto di destinazione B, con una lunghezza di 16 km e della durata di circa 23 minuti in condizioni di traffico normale. Il test è stato ripetuto per vari tragitti con lo smartphone sempre collocato nel porta oggetti centrale tra guidatore e passeggero. Al termine di questo test sono stati effettuate altre registrazioni di tracce dei tragitti posizionando il dispositivo in una posizione casuale. Confrontando i risultati è stato mostrato che gli esiti delle elaborazioni erano molto simili tra di loro, dimostrando che la posizione e l'orientamento del dispositivo all'interno del veicolo è del tutto ininfluente

sul risultato cercato.

Una volta che il sistema ha identificato percorsi significativi, si possono cercare di estrapolare informazioni più dettagliate sul percorso effettuato dall'utente, ad esempio per un utente si può dedurre quale sia il percorso effettuato più frequentemente, le sue abitudini e i posti maggiormente frequentati. Informazioni nel del tutto banali e che, nelle mani sbagliate, potrebbero minare la sicurezza dell'utente.

Lo studio di Michalevsky ([4]) è in grado di capire, tramite il consumo della batteria, la distanza che si interpone tra il dispositivo ed una base che emana le onde radio. Questo è possibile a causa dell'enorme quantità di energia che viene utilizzata quando un segnale radio ha delle interferenze che possono essere causate da vari fattori tra cui alberi ed edifici causando una forte perdita del segnale; infatti, nel tentativo di ristabilire una comunicazione viene impegnata una grande quantità di energia. Al contrario, se il telefono si trova più vicino alla base il consumo di energia del telefono diminuirà. Il problema principale che si può verificare è che i dati sul consumo energetico del telefono sono estremamente aleatori a causa della moltitudine di componenti e applicazioni che consumano contemporaneamente energia. Tuttavia, utilizzando algoritmi di apprendimento automatico si è in grado di dedurre con una buona probabilità di successo la posizione del telefono.

Per l'esperimento sono stati usati tre tipi di obbiettivi di tracciamento degli utenti:

- Distinguibilità del percorso: si cerca di capire se un attaccante può indovinare che strada percorrerà l'utente davanti a un insieme di possibili percorsi.
- tracciamento del movimento real-time: assumendo che l'utente percorrerà una strada conosciuta, si cerca di capire se un attaccante sia in grado di riconoscere la posizione dell'utente lungo il percorso in tempo reale.
- Deduzione di nuove strade: supponendo che l'utente stia percorrendo un lungo percorso, ci si chiede se un attaccante riesca a riconoscere il percorso attraverso le

misurazioni del consumo di energia eettuate in precedenza su molti percorsi più corti nella stessa area.

Per capire inoltre la ripetibilità delle misurazioni del consumo di energia, sono stati effettuati dei test anche sulla potenza del segnale di ricezione: si è infatti notato che percorrendo due volte lo stesso percorso il segnale presenta lo stesso andamento nelle stesse posizioni. Successivamente è stato mostrato che le misurazioni del consumo di energia rivelano un unico pattern stabile per un determinato percorso. Contrariamente alla potenza del segnale, il consumo di energia tende a essere meno regolare perché inuenzato dalla reazione del modem ad un cambio di potenza del segnale. Inne è stato dimostrato, attraverso test in cui sono stati utilizzati differenti modelli di smartphone, che il profilo di consumo di energia rimane stabile a prescindere dal dispositivo usato per la raccolta dei dati. Con un discreto numero di applicazioni attive durante la raccolta dei dati, i risultati ottenuti si possono definire molto accurati su 2/3 degli scenari analizzati, mentre attivando applicazioni di messaggistica istantanea e di social-network il consumo della batteria aumenta in modo notevole e la precisione scende drasticamente a 1/5.

Il contesto di ricerca nel quale si inserisce il presente lavoro è quello di sistemi di navigazione DRS (Dead-Reckoning Systems), ovvero sistemi che elaborano il movimento dell'utente sfruttando letture dai sensori inerziali (accelerometro, giroscopio, magnetometro) al fine di dedurne la posizione esatta. Alcuni studi (ad esempio [13] e [14]) hanno affrontato questa problematica intercalando letture dei sensori inerziali con letture (più sporadiche) del sensore GPS. I primi infatti soffrono di una precisione inferiore rispetto al GPS e possono essere soggetti a rumori interferenziali che non sempre permettono al sensore di produrre un dato realistico, ma hanno un minore consumo di energia. I secondi hanno un'ottima precisione (quanto meno negli ambienti esterni) ma anche un consumo di energia molto maggiore. Unendo, quindi, i pregi di questi tipi di sensore e usandoli per tamponarne le rispettive debolezze si può globalmente aumentare la qualità delle prestazioni di navigazione.

Capitolo 2

Motivazioni

In questo capitolo viene discussa le motivazioni che hanno portato alla concretizzazione dell'elaborato, inoltre verrà illustrato e descritto nel dettaglio il modello utilizzato in questo studio, in particolare uno script scritto in Python([18]) che ha permesso la creazione di vari test ed esperimenti. Uno degli aspetti fondamentali che differenzia questo elaborato dai precedenti è il modo in cui viene affrontato il nucleo del problema, ovvero l'algoritmo di ricerca dei percorsi all'interno di un grafo. In questo caso, dall'intero grafo della città considerata vengono estratti tutti i possibili sottografi che rispettino le seguenti condizioni:

- Il numero di segmenti deve essere uguale a quanto specificato da un parametro di input.
- Gli angoli di curvatura tra le rispettive coppie di segmenti deve anche corrispondere ad una serie di valori di input.

A questo punto la ricerca dei percorsi viene condotta esclusivamente su tali sotto grafi applicando degli opportuni criteri di corrispondenza.

In questo elaborato si presentano anche le fasi di simulazione ed analisi. La simulazione viene effettuata tramite delle funzioni che verranno descritte in seguito, generando dei percorsi possibili su una mappa reale e introducendo vari gradi di rumore sulle misurazioni. In questo modo viene simulato lo spostamento di un veicolo lungo un percorso

stradale verosimile.

Infine viene effettuata l'analisi sui dati ottenuti con opportune considerazioni.

I moderni dispositivi mobili offrono un gran numero di servizi, molti dei quali basati sulla localizzazione dell'utente. In genere il telefono cellulare fa riferimento al sensore GPS per calcolare la posizione del dispositivo. Il livello di precisione ottenibile, basato sulla triangolazione di segnali satellitare, è piuttosto elevato. Si immagini il rischio connesso alla protezione della privacy di un utente se tali dettagli della posizione venissero trafugati da un'applicazione maliziosa: seguire i movimenti di un utente, analizzarne le abitudini ed individuarne le posizioni maggiormente frequentate sono attività che potrebbero generare addirittura delle condizioni di pericolo, oltre ad una violazione della privacy.

Queste considerazioni sono ben note agli sviluppatori dei sistemi operativi mobili, come per esempio Android; come precedentemente evidenziato, tali sistemi richiedono l'autorizzazione esplicita dell'utente prima di poter accedere alle rilevazioni del sensori. Negli Smartphone moderni il GPS è soltanto uno dei tanti sensori che possono trovarsi al suo interno; per citarne alcuni, sono presenti sensori inerziali quali magnetometri, accelerometri e giroscopi, tutti sensori che non sono protetti da nessuna barriera di autorizzazioni: una qualunque applicazione può accedervi senza che l'utente possa saperlo.

In questo elaborato si vuole mostrare la possibilità di rintracciare la posizione degli utenti tramite l'aiuto dei soli sensori sopra citati e quindi senza chiedere alcuna autorizzazione. Nel nostro studio si vuole evitare l'utilizzo del GPS (tra l'altro caratterizzato da un dispendioso consumo di energia) e dimostrare che tramite i soli sensori inerziali si può arrivare ad un'accuratezza di localizzazione paragonabile a quella tipica del GPS.

L'idea generale consiste nello sfruttare l'accelerometro e il magnetometro:

- l'accelerometro per stimare la lunghezza di un segmento di strada percorso
- il magnetometro per rilevare l'angolo di sterzata e quindi comprendere quando il dispositivo si è mosso da un segmento all'altro del tragitto.

Tali segmenti sono assimilabili al concetto di grafo (composto da archi e nodi); da un'analisi di questi grafi e da un confronto con le mappe scaricate da OpenStreetMap si può risalire (talvolta con un'ottima precisione) al tragitto reale percorso dall'utente.

Capitolo 3

Sviluppo

La parte fondamentale di questo elaborato riguarda principalmente la creazione e l'utilizzo di sottografi, elemento utilizzato in sostituzione del grafo della città presa in considerazione. Ciò viene attuato attraverso lo sviluppo di uno script Python; più precisamente saranno descritti i vari passaggi implementati al fine di riuscire a trovare nel modo migliore il percorso simulato all'interno del grafo. Infine verrà descritta una problematica riscontrata in corso d'opera e la soluzione adottata.

3.1 Creazione del grafo

Il primo passo consiste nel costruire il grafo relativo alla città oggetto delle simulazioni. Questo ci consente di sfruttare librerie e algoritmi su grafi per automatizzare la ricerca. Per affrontare questo passaggio si fa affidamento su alcuni pacchetti disponibili per il linguaggio di programmazione Python, in particolare verranno esaminate: NetworkX[19] e OSMnx[20].

3.1.1 Architettura del sistema

Nella Figura 3.1 viene raffigurata graficamente l'architettura del sistema. Quest'ultimo si sviluppa nei seguenti passaggi:

- L'utente interagisce con il sistema di elaborazione mandando una richiesta all'interno della quale vengono viene specificata la città ed i valori di "Length" e "Bearing" da analizzare.

- Il sistema riceve la richiesta ed interroga lo storage verificando la presenza dei sottografi della città richiesta. Nel caso in cui la richiesta effettuata abbia un esito positivo il sistema esporta i dati dallo storage, li elabora e restituisce il risultato all'utente. Se invece la richiesta riscontri un esito negativo, il sistema interpella OSMnx.

- OSMnx, una volta ricevuta la richiesta, crea il grafo della città e lo invia al sistema; quest'ultimo elabora il grafo, crea tutti i sottografi della città, li salva nello storage ed lo invia l'elaborato all'utente.

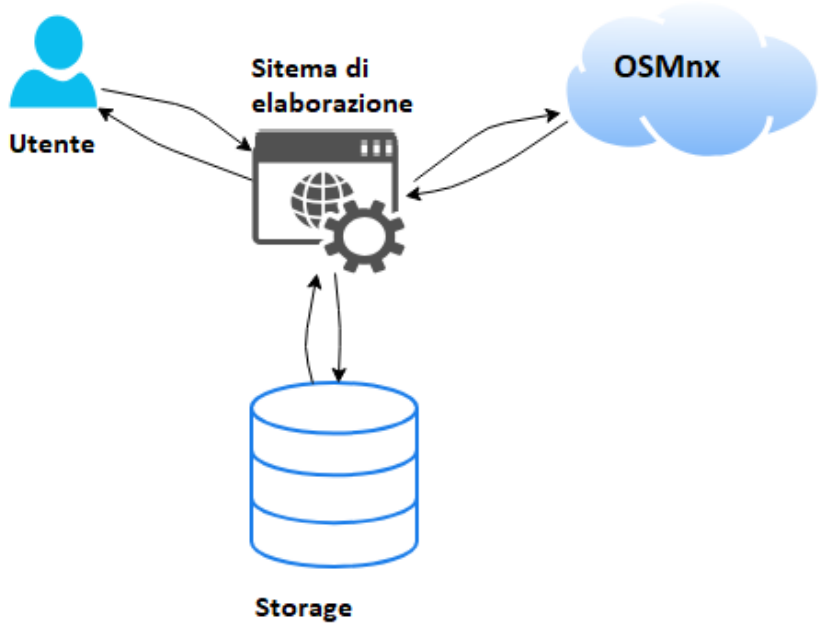

Figura 3.1: Architettura del sistema

3.1.2 NetworkX

NetworkX è un pacchetto per la creazione e la manipolazioni di strutture dati come grafi ed alberi. Al suo interno sono disponibili diverse funzioni che possono essere usate per operazioni meno complesse come la creazione di un nodo o di un arco, aggiunta dei valori agli archi fino ad arrivare all'implementazione di algoritmi più sofisticati come possono essere Algoritmo di Dijkstra[21], algoritmo ricerca in ampiezza(BFS[22]).

3.1.3 OSMnx e la sua struttura

Il pacchetto in questione consente di costruire, progettare, visualizzare e analizzare facilmente reti stradali complesse in Python tramite l'aiuto di NetworkX. OSMnx è stato scelto perché ha quattro importanti qualità:

- Riesce in modo completamente automatico a carpire i confini di una città.
- Costruzione automatizzate e su misura dei dati della rete stradale da OpenStreetMap.
- La correzione algoritmica della topologia di rete.
- La capacità di analizzare le reti stradali, compreso il calcolo di percorsi, la proiezione e la visualizzazione di reti e il calcolo di misure metriche e topologiche

Una volta inserita la città da analizzare il pacchetto creerà il grafo richiesto come visibile nella figura sottostante 3.2, il quale sarà un grafo non orientato contente informazioni su strade ed incroci i quali saranno analizzati nell'algoritmo di confronto.

3.2 Simulazione dei percorsi

Una delle prime funzionalità implementate in Python è quella per la generazione di percorsi simulati. Queste funzioni hanno il compito di costruire il grafo dettagliato di un percorso che potrebbe essere verosimilmente svolto da un utente. Negli algoritmi realizzati, archi e nodi del grafo sono caratterizzati dai seguenti parametri:

- "Length": la lunghezza dei segmenti della strada;

Figura 3.2: Grafo della città di Modena

- "Bearing": l'angolazione di curvatura tra un segmento e l'altro.

Tali parametri vengono estratti attraverso una ricerca su grafo di mappa reale, in modo che i percorsi simulati siano effettivamente corrispondenti ad uno scenario realistico.

3.2.1 Estrazione lunghezza dei segmenti simulati

Per quanto riguarda la creazione dei segmenti simulati, vengo presi due nodi in modo casuale dal grafo corrispondente alla città designata per la simulazione; tali nodi vengono elaborati da una funzione la quale restituisce un singolo elenco contente tutti i nodi

intermedi dal nodo d'origine al nodo di destinazione, secondo un criterio di cammino minimo. Questa operazione sarà ripetuta il numero di volte necessario ad ottenere i grafi di simulazione aventi una certa dimensione voluta. Per la ricerca dei cammini minimi all'interno di un grafo viene impiegato l'uso della libreria descritta nel sotto paragrafo 3.1.2 più precisamente viene adoperata la funzione chiamata"shortest_path".

```
    while not found:
        try:
            node1 = np.random.choice(g.nodes())
            node2 = np.random.choice(g.nodes())
            route = nx.shortest_path(g, node1, node2)
            if len(route) in lunghezzaPercorsoDaInserire:
                found = True
        except:
            pass
    return node1, node2
```

Listing 3.1: Codice per il recupero delle lunghezze

Una volta trovato un percorso della lunghezza corretta, il passo successivo è scansionare un dizionario globale in cui sono memorizzati gli archi con i relativi valori delle lunghezze dei segmenti, ed infine salvare i valori corretti su di una lista.
Questa lista verrà restituita come parametro di output.

```
    for item in route:
        if previous == -1:
            previous = item
            continue
        try:
            path.append(length_g[(previous, item, 0)])
        except:
            print("Not found")
        previous = item
    return  path
```

Listing 3.2: Recupero Informazioni inerenti alla Length

3.2.2 Estrazione delle angolazioni

Il funzionamento inerente alle estrazioni delle angolazioni si comporta in modo del tutto simile alla sezione 3.2.1 fatta eccezione per due aspetti:

- I nodi non vengono scelti in modo casuale ma sono gli stessi di quelli usati per il grafo dal quale sono state estratte le lunghezze;
- Dal dizionario globale vengono estratti dati riguardanti le angolazione dei percorsi.

Una volta richiamate queste funzioni si avrà il percorso simulato composto dalle lunghezze dei segmenti e dalle angolazioni delle strade pronti per essere confrontati con quelle prodotte dall'algoritmo che rileva il movimento dell'utente.
Tale algoritmo verrà descritto in modo dettagliato nei prossimi paragrafi.

3.3 Creazione dei sotto grafi

In questa sezione verrà trattato uno degli algoritmi che ha caratterizzato questo lavoro e lo rende differente dagli altri elaborati. Per analizzare il problema in un modo diverso si è deciso di implementare un algoritmo che gestisca la creazione di tutti i possibili sottografi della città analizzata. La dimensione dei sottografi può variare e viene decisa in base al numero di curve e segmenti di lunghezze prodotte dalle funzioni descritte nei sotto paragrafi 3.2.1 e 3.2.2.

Analizzando nello specifico l'algoritmo, come primo step si ha bisogno di scorrere il grafo principale in modo tale da analizzare, un nodo alla volta, la città considerata; ogni nodo sarà assegnato come parametro di ingresso ad una nuova funzione creata con lo scopo di trovare tutti i vicini del nodo passato dalla funzione chiamante; questa azione sarà ripetuta ricorsivamente fino alla creazione di un sotto grafo della dimensione specificata.

```
for inizioNodi in g.node():
    prog +=1
    for path in simple_paths(inizioNodi, len(lengths)):
```

```
        subPath.append(path)

```

Listing 3.3: Funzione Chiamante

Una volta trovato un sotto grafo della dimensione corretta lo si restituisce come output al chiamante, sospendendo l'esecuzione in modo temporaneo della funzione (viene effettuato tramite l'uso della funzione "yield" di Python), mantenendo intatto lo stato di esecuzione. Non appena la funzione chiamante effettua l'operazione di scrittura del sottografo trovato, sarà richiamata di nuovo la funzione per la ricerca dei vicini che ripartirà dall'ultimo stato di sistema salvato e progredirà nella creazione di tutti i sotto grafi possibili.

```
    if length==0:
        yield(visitato + [inizio])
    else:
        for figlio in nx.neighbors(g,inizio):
            if figlio not in visitato:
                for path in simple_paths(figlio, length-1, visitato + [inizio]):
                    yield(path)
```

Listing 3.4: Funzione chiamata

Ripetendo l'esecuzione per il numero di nodi del grafo si avrà come output l'insieme totale dei sottografi. Per fare in modo di non dover ricalcolare sempre i sotto grafi già computati precedentemente (ovvero dove numero di curve e lunghezze coincidono) si è deciso di implementare una scrittura sul file per salvare l'intera struttura dati riguardante i sotto grafi calcolati. All'occorrenza il contenuto di tale file può essere letto e ricreato in memoria.

3.4 Unione dei nodi con la medesima angolazione

Per preparare lo script ad un contesto reale si è cercato di prevedere gli scenari di errore che si possono verificare durante i test e che è necessario gestire. Nel nostro lavoro è

necessario gestire l'errore di misura a cui sono soggetti i sensori inerziali ed interpretarne in maniera corretta le rilevazioni. Tutto questo tenendo conto della logica implementata nell'applicazione Android sviluppata per raccogliere informazioni dai sensori.

Nell'implementazione pratica, l'applicazione prende in considerazione nuovi segmenti tramite l'accelerometro fino a quando il dispositivo mobile cambia orientamento, ovvero è stata eseguita una sterzata. Si supponga che l'utente stia guidando su una strada diritta ed attraversa uno o più incroci senza però sterzare (mantenendo la traiettoria corrente), In questo scenario l'applicazione non è in grado di rilevare la presenza dei nodi relativi agli incroci attraversati; bensì la porzione di rotta percorsa in linea retta viene riconosciuta come un unico segmento di grafo.

Questo rappresenta un ostacolo per il corretto riconoscimento del percorso all'interno del grafo cittadino. Infatti la libreria che effettua il parsing della topologia della città in forma di grafo crea un nodo per ogni incrocio rilevato, non avendo a priori conoscenza della modalità con cui l'incrocio sarà percorso.

Per ovviare a questo problema è stata implementata una funzione che riadatta tutti i sotto grafi computati. Più nel dettaglio il primo passo svolto dalla funzione è controllare che nel sottografo da analizzare si trovino dei valori di "Bearing" (valore di angolazione di una curva) adiacenti ed uguali. Nel caso venissero rivelati i medesimi valori si invoca un secondo algoritmo in grado di produrre un nuovo percorso nel grafo e che va a sostituire tutti i nodi intermedi dove è presente l'uguaglianza e l'adiacenza, ed infine unire questi nodi creando un singolo segmento la cui lunghezza è data dalla somma di tutte le lunghezze dei nodi eliminati. Quanto detto fino ad ora viene riportato graficamente nella Figura 3.3 che rappresenta lo stato del grafo prima dell'esecuzione dell'algoritmo fino ad arrivare alla Figura 3.4 che ne riporta la situazione finale dopo l'esecuzione.

Figura 3.3: Porzione di grafo prima dell'unione dei nodi

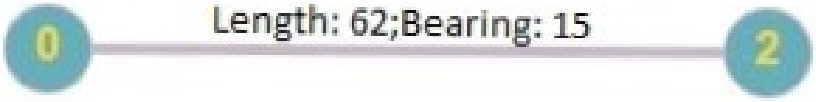

Figura 3.4: Porzione di grafo unita

3.5 Algoritmo di ricerca dei sotto grafi

Prima di illustrare il passo successivo e cioè l'algoritmo per l'effettiva ricerca di un percorso tra i sottografi di una città, si riassumono per comodità nella Figura 3.5 i passaggi logici esposti nei paragrafi precedenti.

- (a): esempio di percorso simulato estratto dal grafo della città (vedere sotto paragrafo Simulazione dei Percorsi).
- (b): la rielaborazione in forma di grafo della città presa in considerazione.
- (c): tutti i sotto grafi computati dall'algoritmo presente nella sezione 3.3 di questo elaborato. Si noti che la lunghezza di questi sottografi corrisponde a quella del percorso da ricercare.

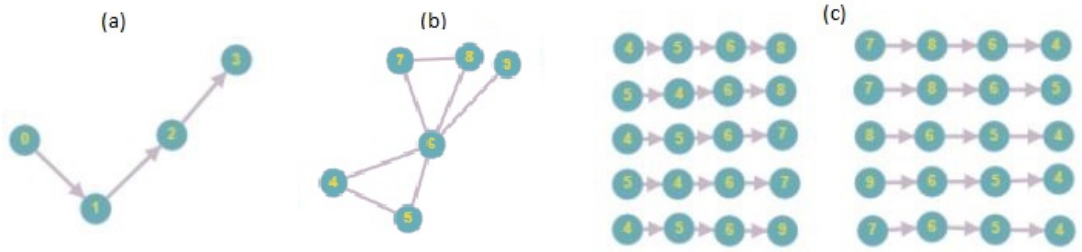

Figura 3.5: Riassunto grafico

Si ricorda che in questa prima fase di sperimentazione i percorsi di test sono completamente simulati, scelti dal grafo della città con un certo criterio di lunghezza. Come tali i percorsi di test sono "ideali", ovvero puliti dai rumori dei sensori. In condizioni reali, invece, sono diverse le motivazioni che possono portare ad avere rilevazioni sensoriali imperfette (il cosiddetto rumore dei sensori). Alcune delle cause che comportano questi errori sulle misure sono:

- terreni sconnessi;
- stile di guida più o meno regolare degli utenti;
- errori interni sull'allineamento dei sensori;
- temperatura del dispositivo, che può creare scostamenti numerici dai valori reali.

Per ottenere dei risultati di simulazione più interessanti, si aggiunge artificialmente un certo livello di "rumore" sui dati di percorso. Ovviamente maggiore sarà il rumore e peggiore sarà la qualità del matching, e maggiori i percorsi trovati. Per l'aggiunta del rumore si definisce un intervallo di valori (in questo lavoro sono stati usati valori di 5, 10, 20, 30) che vengono aggiunti o sottratti dalle lunghezze dei segmenti e/o dai valori di curvatura dei nodi.

Analogamente si definisce una soglia entro la quale accettare come valido un certo grafo rispetto al percorso di input; al variare di questa soglia è possibile valutare le diverse prestazioni dell'algoritmo che si descrive nel seguito.

L'algoritmo che fa il "match" tra il percorso simulato e tutti i sotto grafi presenti nella lista sfrutta i valori di "Bearing" e "Length" sia dei percorsi simulati che dei sotto grafi computati.
Il corpo della funzione analizza il sottografo del percorso preso in considerazione e ne analizza le sequenze di "Bering" e "Length". Una volta aggiunto il rumore si effettua un confronto con i valori di length e bearing dei percorsi simulati ricevuti come input alla funzione; se rientrano nel rangedi accettabilità allora il percorso viene considerato valido ed inserito un una lista di percorsi candidati.

```
 for leng, ber in zip(lengths, bearings):
     if i >= 0 and j >= 2 and flag == 0:
         nodeTT = path[i:j]
         nodeBearing = ox.get_route_edge_attributes(g, nodeTT, 'bearing')
         nodeLength = ox.get_route_edge_attributes(g, nodeTT, 'length')
         if nodeBearing[0] >= ber - int(rumoreBearing) and nodeBearing[0] <= ber + int(rumoreBearing):
             if nodeLength[0] >= leng - int(rumoreLength) and nodeLength[0] <= leng + int(rumoreLength):
                 i += 1;
                 j += 1
                 flag = 0
                 tt=True
                 if (len(lengths)) == i:
                     pathGiusta.append(path)
             else:break
         else:break
```

Listing 3.5: Calcolo soluzione finale

Ripetendo questo passaggio per tutti i sotto grafi computati, alla fine di questa funzione si avrà una lista di tutti i possibili percorsi che risultano essere simili a quello generato per la simulazione. La Figura 3.6(d) aggiunge questo ulteriore passaggio logico, affiancandosi a tutti quelli precedenti già descritti.

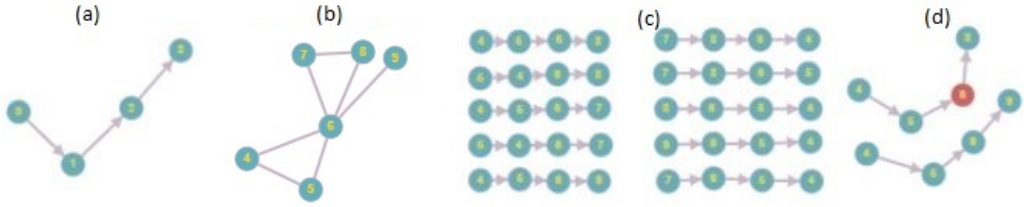

Figura 3.6: Riassunto grafico

3.6 Raffinare i risultati di ricerca

Come anticipato nel paragrafo precedente l'algoritmo di matching, per effetto del rumore dei sensori e della soglia di tolleranza sulla valutazione del risultato, porta ad avere una lista di risultati verosimili. In questo paragrafo si presenta un metodo per perfezionare i risultati ottenuti in modo da avvicinarsi il più possibile ad un risultato esclusivo.

Nella funzione vengo definite due matrici, rispettivamente chiamate 'A' ed 'L', per la valutazione degli scostamenti numerici tra le soluzioni dell'algoritmo di match e il grafo di input. In particolare:

- La prima matrice 'A' contiene al suo interno le differenze del valore di angolazione della curva presente nei risultati elaborati nella sezione 3.5, ed il valore di angolazione della curva inseriti come input.
- La matrice 'L' viene calcolata allo stesso modo ma la differenza viene effettuata sulla lunghezza dei segmenti.

Nell'analisi dello scostamento non è importante il segno della differenze, quindi per eliminare la dipendenza dal segno viene utilizzato il valore assoluto. Dopo i passaggi di creazione delle matrici vengono effettuate le somme dei valori interni, e si definiscono due coefficienti di peso "Alpha" (α) e "Beta" (β) tali che

$$\alpha + \beta = 1 \tag{3.1}$$

Tali coefficienti vengono adoperati per dare più o meno importanza alla matrice delle lunghezze ovvero a quella delle curvature. Chiaramente i valori di α e β hanno una forte influenza, metterli uguali tra loro significa che ci preoccupiamo delle lunghezze dei segmenti e delle angolazioni delle curve allo stesso modo. Aumentare α dà più importanza all'angolazione della curva, mentre diminuisce l'importanza della lunghezza dei segmenti e aumentando β si verifica il processo opposto. Questi due coefficienti vengono infine aggregati linearmente in un indicatore σ ("sigma") che rappresenta una sorta di punteggio di errore relativo ad ogni grafo ottenuto dal matching. Valori di σ inferiori indicano match migliori; il minimo valore di σ indica il miglior risultato tra quelli ottenuti.

```
    for leng, ber in zip(lengths, bearings):
        nodeTT = path[i:j]
        nodeBearing = ox.get_route_edge_attributes(g, nodeTT, 'bearing'
    )
        nodeLength = ox.get_route_edge_attributes(g, nodeTT, 'length')
        if(A.item(z,y)==0):
            A[z][y] = np.absolute(nodeBearing[0] - ber)
            A[y][z] = np.absolute(nodeBearing[0] - ber)
        if(L.item(z,y)==0):
            L[z][y] = np.absolute(nodeLength[0] - leng)
            L[y][z] = np.absolute(nodeLength[0] - leng)
        i = i + 1;z= z + 1
        j = j + 1;y= y + 1
    sumBer=A.sum()
    sumLeng=L.sum()
    sigma=(alpha*sumBer)+(beta*sumLeng)
```

Listing 3.6: Raffinare i risultati

Per determinare i valori appropriati di α e β viene calcolato il massimo ed il minimo di una Probability Density Function (PDF); una tale funzione di distribuzione della probabilità, come meglio evidenziato nel prossimo capitolo, può essere considerata una caratteristica peculiare di ogni città considerata. Sottraendo il minimo al massimo ricaviamo lo scarto tra i due valori. Ricavando la media della PDF siamo in grado di

determinare due segmenti che, normalizzati rispetto al valore dello scarto massimo, ci restituiscono i valori di α e β (come lunghezza dei due segmenti in cui è stato diviso il segmento di partenza dal valore della media). In questo modo se i valori delle angolazioni risultano essere simili allora la media dei valori comporterà due parametri di simile entità, in quanto il segmento dato dallo scarto tra il minimo della PDF ed il suo massimo verrà tagliato approssimativamente al centro. Se invece i parametri presentano dei picchi nettamente accentuati (come ad esempio avviene in città dove sono presenti strade strutturate con molte svolte con un angolo di 90) il valore medio sarà spostato verso il basso ricavando così valori dei due parametri molto diversi.

Nella Figura 3.17 viene rappresentato graficamente quanto detto; il grafico rappresenta la città di Milano Alpha e Beta in questo caso assumono un valore molto simile tra loro ($\alpha = 0.4$, $\beta = 0.6$). Nella figura 3.8 viene rappresentata la città di Miami, caratterizzata da una forma a scacchiera con la maggior parte delle curve a 90; in questo caso i valori di Alpha e Beta saranno molto distanti tra loro ($\alpha = 0.107327435$, $\beta = 0.892672565$). Il problema riportato in precedenza sarà discusso in modo più specifico nel capitolo successivo. La Figura 3.9 ha lo scopo di riassume graficamente tutti i passaggi compiuti nel Capitolo 3.

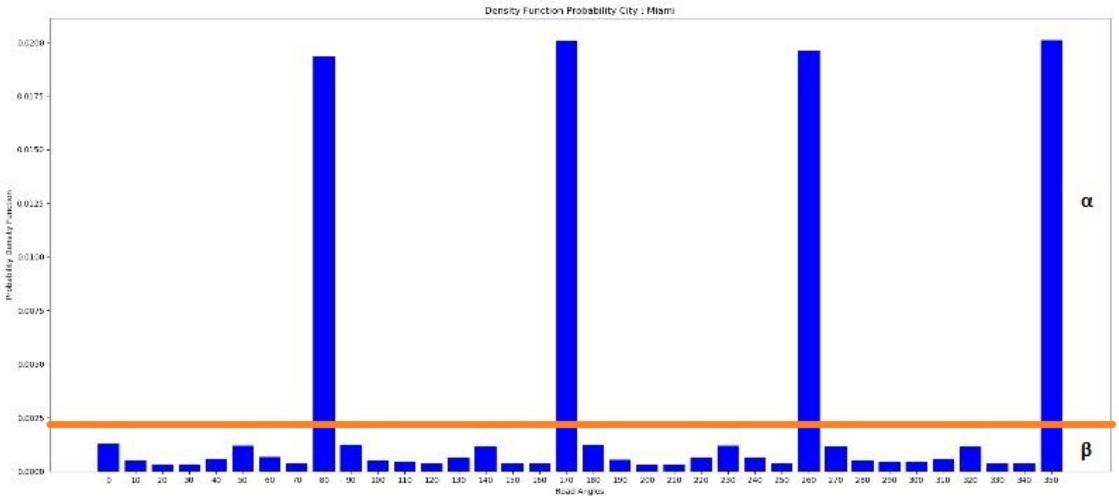

Figura 3.7: Grafico rappresentante la città di Miami

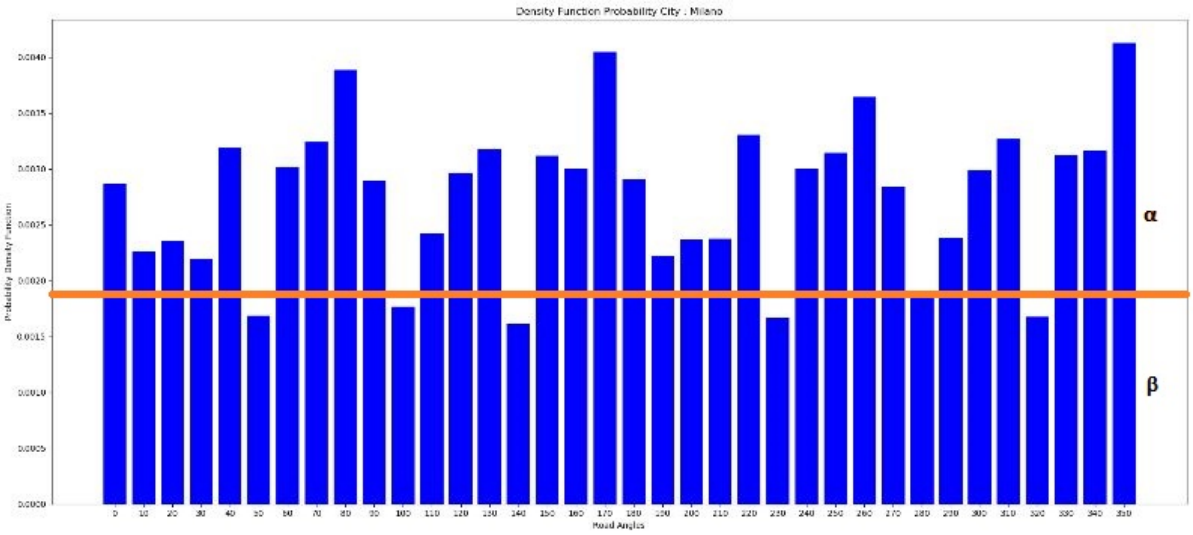

Figura 3.8: Grafico rappresentante la città di Milano

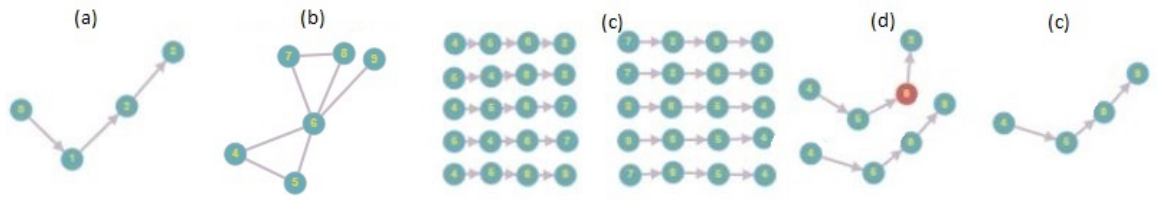

Figura 3.9: Schema completo dei procedimenti effettuati

3.7 Creazione dei file di Output

Ogni qual volta lo script python viene eseguito, tutti i risultati utili allo svolgimento di un analisi che verrà affrontata nel capitolo successivo vengono raccolti in una serie di file CSV (Comma-Separated Values - Valori separati da virgola). I dati salvati vengono utilizzati principalmente per lo studio dei comportamenti di diverse città utile a capire i diversi errori che si potrebbero verificare al fine di evitarli anche città dalla topologia simile.

Capitolo 4

Analisi e risultati finali

In questo capitolo si descriveranno le analisi effettuate sui dati raccolti dalle simulazioni e saranno discussi i risultati ottenuti. L'analisi è stata effettuata tramite dei grafici che verranno descritti in seguito, facendo notare le varie differenze che incorrono tra le città di diversi continenti. Per queste analisi sono state prese in considerazione due continenti, Europa ed America. In particolare:

- Per quanto riguarda l'Europa, in particolare l'Italia, le città analizzate sono: Bologna, Milano.
- Per L'Europa in generale sono state scelte città come: Nizza e Dublino.
- Sono state prese in considerazione per L'America città come: Manhattan Miami, Boston.

4.1 Implementazione dei grafici

Per la creazione dei grafici in Python viene usata Matplotlib([23]) (una libreria Python per la creazione di grafici). Ad essa viene affiancata una libreria matematica di supporto alle elaborazioni necessarie; tra le varie alternative è stata scelta la libreria NumPy. NumPy è una libreria open-source con delle interessanti caratteristiche: aggiunge supporto a grandi matrici e array multidimensionali oltre ad una vasta collezione di funzioni

matematiche di alto livello per poter operare efficientemente su queste strutture dati (strutture nativamente non disponibili in Python).

Per quanto riguarda il funzionamento degli script per la creazione dei grafici uno dei primi passaggi che viene effettuato è quello di importare i dati dei file CSV che sono stati creati tramite l'esecuzione dello script "subGrapy". I dati importati verranno salvati su di una semplice lista Python. Ed infine la lista sarà processata dalla libreria per la creazione dei grafici

4.2 Studio preliminare dei grafi

Per cercare di ricreare un contesto che si avvicini maggiormente a tutte le possibili casistiche che si possono verificare nella realtà, sono state prese in considerazione città di diverse grandezze. Quando si parla di grandezza della città in questo contesto non si fa riferimento all'area di superficie occupata o alla densità di popolazione, bensì si fa riferimento al numero di nodi e di archi che compongono il grafo della città presa in considerazione.

Nella Figura 3.1 vengono riportati graficamente per ogni città studiata il numero di nodi (indicati con il colore azzurro) che l'algoritmo deve processare prima di arrivare ad avere un intera lista di sottografi. Mentre gli archi del grafo sono identificati tramite l'uso del colore arancione. Come si può notare i numero di nodi che sono presenti per Nizza, Bologna, Dublino e Manhattan sono più o meno equivalenti e non superano la quota di 6000; invece città come Milano, Boston e Miami arrivano ad un picco massimo che è quasi al doppio dei nodi delle tre città precedenti. Per quanto riguarda gli archi del grafo, le città sembrano avere un andamento similare (relativamente al rapporto tra nodi ed archi) tranne per quanto riguarda la città di Boston dove il numero di archi triplica rispetto al numero dei nodi.

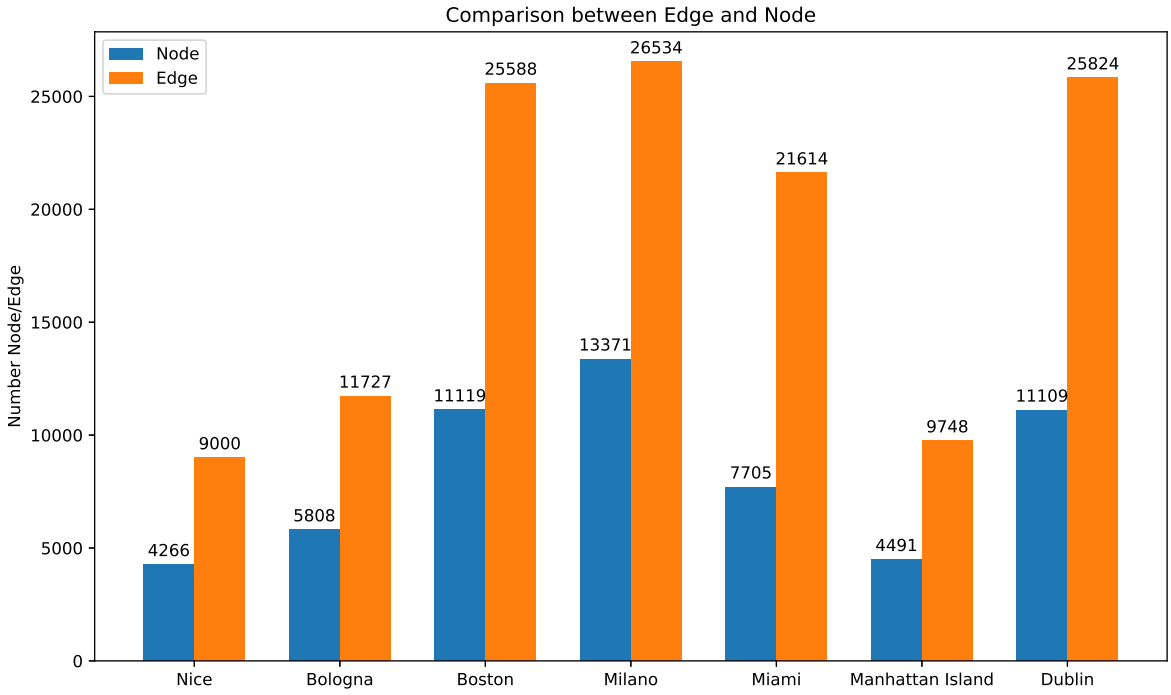

Figura 4.1: Grafico dei nodi e degli archi delle città

4.3 Studio della densità di probabilità

Funzione di densità di probabilità (o PDF) è la funzione f(n)=R→ R che ad ogni reale associa il limite per dx che tende a 0, del rapporto tra la probabilità che la variabile casuale assuma valori nell'intervallo (x, x + dx] e l'ampiezza dx.
La definizione viene riportata in simboli matematici :

$$f(x) : R \rightarrow R : \tag{4.1}$$

$$x \rightarrow lim_{dx \rightarrow\ 0} = \frac{P(x < X \leq x + dx)}{dx} \tag{4.2}$$

La funzione di densità in x rappresenta quanto vale la probabilità "intorno ad x" in rapporto all'ampiezza di tale "intorno". Il termine funzione di densità serve proprio ad evocare quanto è "densa" la probabilità che un determinato evento avvenga.

Tramite l'uso di quanto descritto sino ad ora (ovvero della funzione PDF) si cerca di effettuare uno studio riguardante il numero di segmenti dei grafi stradali che hanno una densità maggiore e poi gli angoli di curvatura, cercando di evidenziare le maggiori differenze tra le città Europee ed Americane.

4.4 Analisi densità di probabilità delle città

In questa sezione verranno studiate le probabilità di densità delle varie città sopra citate, soffermandosi particolarmente sulle differenze tra le metropoli europee ed americane; tali divergenze sono visibili graficamente sia sotto forma di diagramma che di struttura stradale(dalla Figura 4.2 alla 4.8).

Figura 4.2: Mappa Bologna

Figura 4.3: Mappa Dublino

Figura 4.4: Mappa Nizza

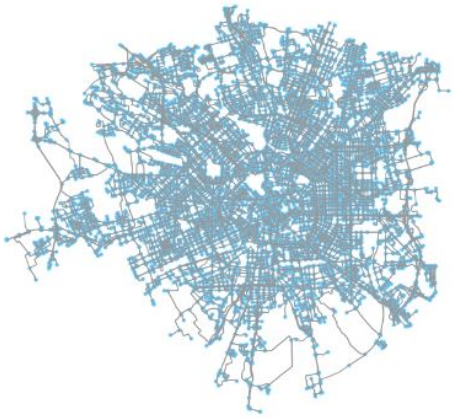

Figura 4.5: Mappa Milano

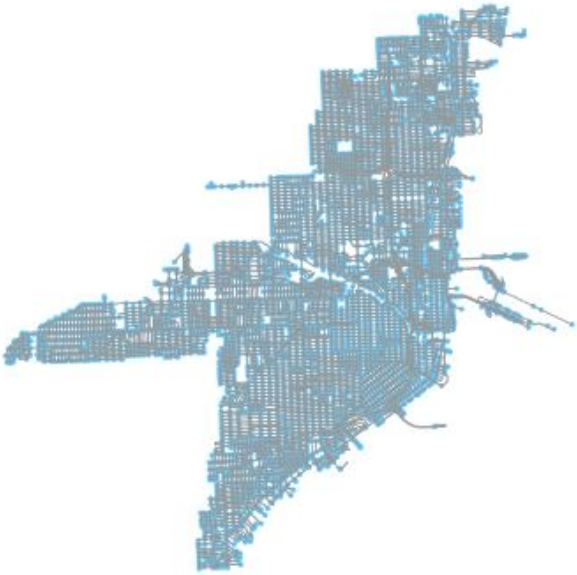

Figura 4.6: Mappa Miami

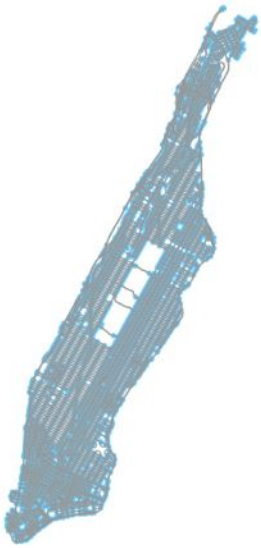

Figura 4.7: Mappa Manhattan

Figura 4.8: Mappa Boston

4.4.1 PDF delle Città Europee

Per quanto riguarda il calcolo della PDF saranno prese in considerazione due città europee, Dublino e, in particolare per le città italiane, Bologna.

4.4.1.1 Città di Bologna

L'analisi parte dalla città italiana di Bologna riportata nella Figura 4.9, in cui viene rappresentata la PDF calcolata sulle lunghezze dei segmenti. Dal grafico compare in modo evidente che il numero di segmenti con maggiore probabilità di essere rinvenuti sono quelli la cui lunghezza va da 0 a 50, da mentre da 50 a 200 la distribuzione riceve un picco in cui si abbassa notevolmente. Al di là di questa soglia si arriva ad un massimo visibile di 900 prima che il grafico abbia andamento costante con bassissima probabilità che si possano trovare segmenti di lunghezza delle misure indicate sull'asse X.

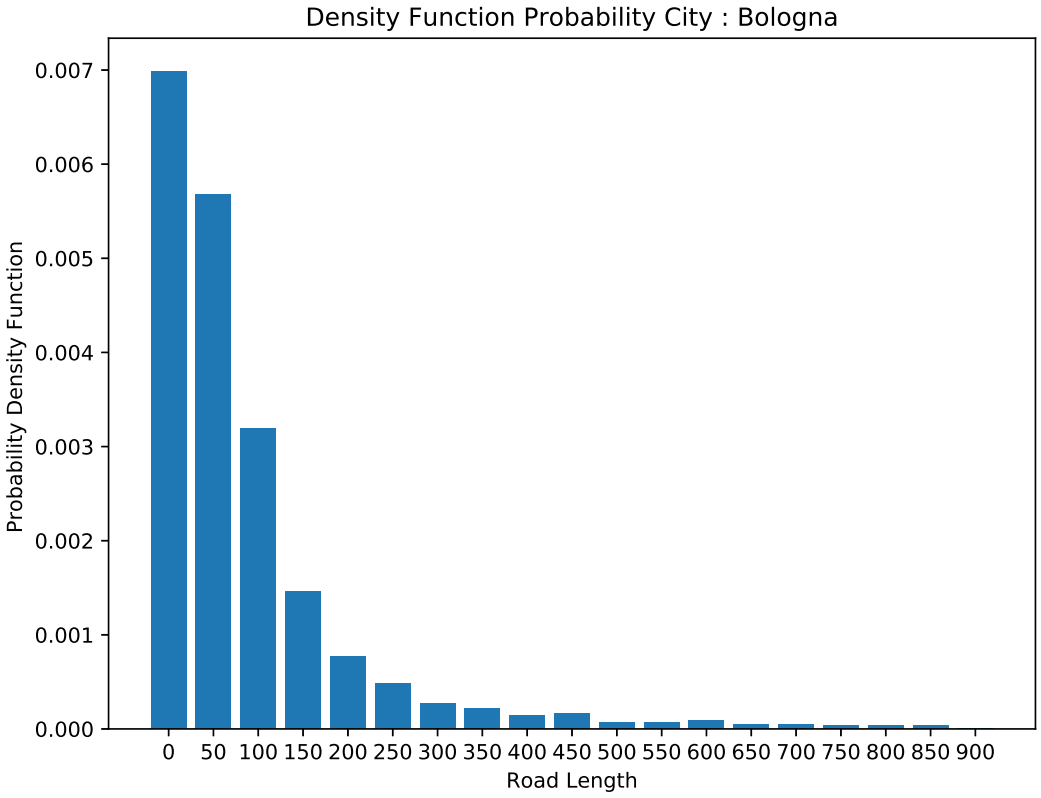

Figura 4.9: PDF di Bologna calcolata su "Length"

Dalla Figura 4.10, in cui viene analizzata la distribuzione degli angoli di curvatura di Bologna, salta subito all'occhio la presenza di una distribuzione più variegata: questo rappresenta un fattore positivo per l'elaborato in questione perché maggiori sono le differenze tra tutti gli angoli di curvatura in una città, più cresce la probabilità di ritrovare un percorso preciso. Si ricorda che questo problema è dovuto essenzialmente al rumore dei sensori: un telefono con sensori più costosi sarebbe in grado di rilevare dati più accurati e dettagliati, con conseguente maggiore capacità di trovare un percorso più corretto. Tuttavia grazie all'analisi dei grafi delle mappe cittadine, si riesce a contrastare efficacemente il problema delle misurazioni disturbate, e questo è di particolare importanza per delle città con una struttura a scacchiera più regolare come quelle americane (analizzate successivamente).

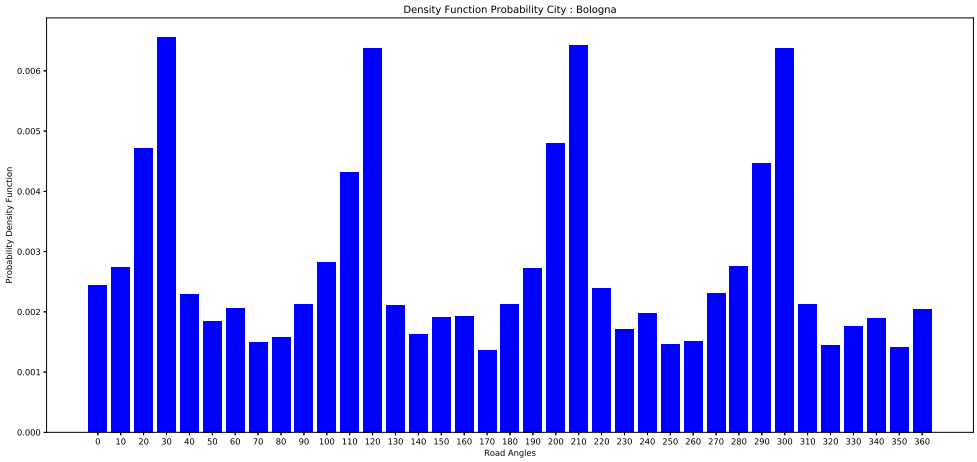

Figura 4.10: PDF di Bologna calcolata sulle Bearing

4.4.1.2 Città di Dublino

La città di Dublino presenta caratteristiche molto simili, per quanto riguarda il profilo della PDF, alla città di Bologna. Come si nota dal grafico di Figura 4.11, le lunghezze dei segmenti di strada che sono maggiormente presenti sono ripartiti da 0 a 100 con il picco massimo a 50; da 150 in poi il numero di segmenti scende notevolmente finché la funzione non raggiunge, anche in questo caso, un andamento asintotico con piccolissimi e sporadici sbalzi.

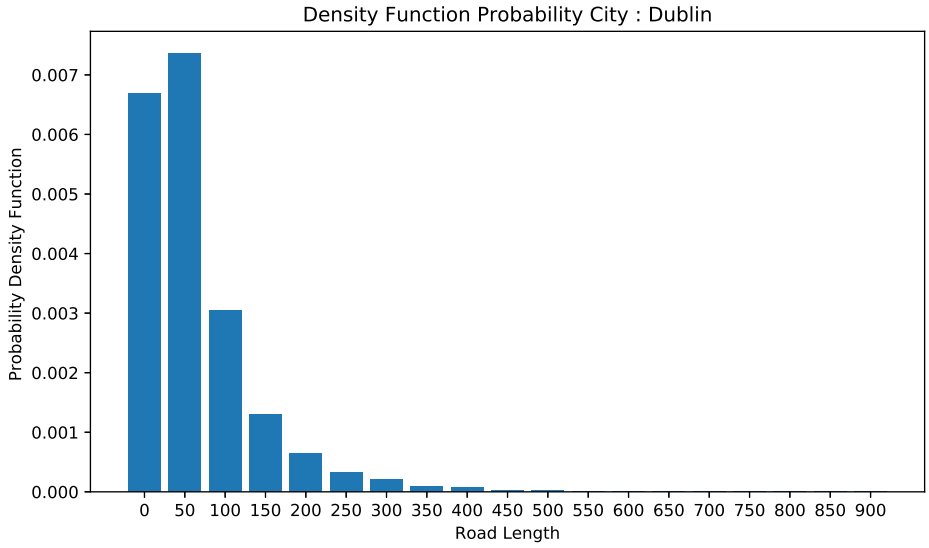

Figura 4.11: PDF di Dublin calcolata sul Length

Per quando riguarda le angolazioni delle curve, evidenziate nella Figura 4.12, anch'esse rispecchiano molto l'andamento riportato per la città di Bologna tranne per il fatto che, in questo grafico, sono riportati picchi meno evidenti.

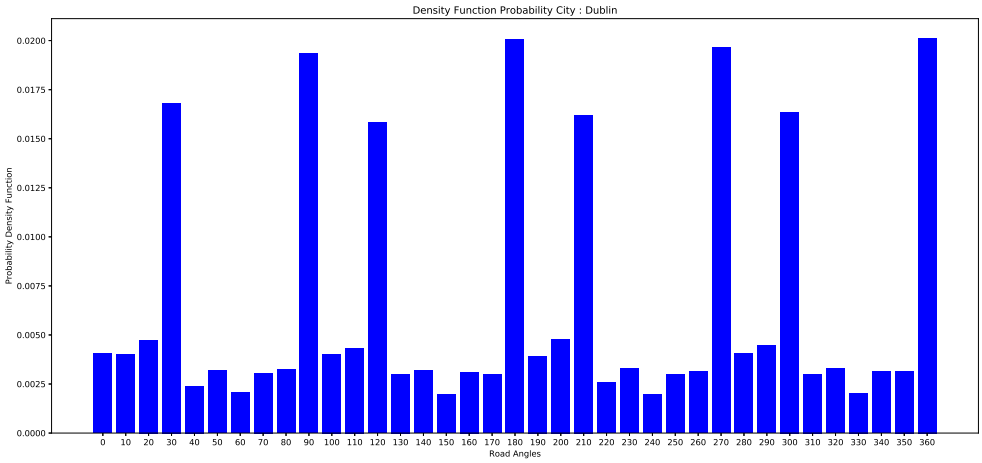

Figura 4.12: PDF di Dublin calcolata sul Bearing

4.4.1.3 Città di Nizza

Dal grafico della città di Nizza si evidenzia le lunghezze dei segmenti di strada che sono maggiormente presenti sono divisi da 0 a 50; da 100 in poi il numero di segmenti scende notevolmente finché la funzione non raggiunge, anche in questo caso un andamento asintotico tendente allo 0.

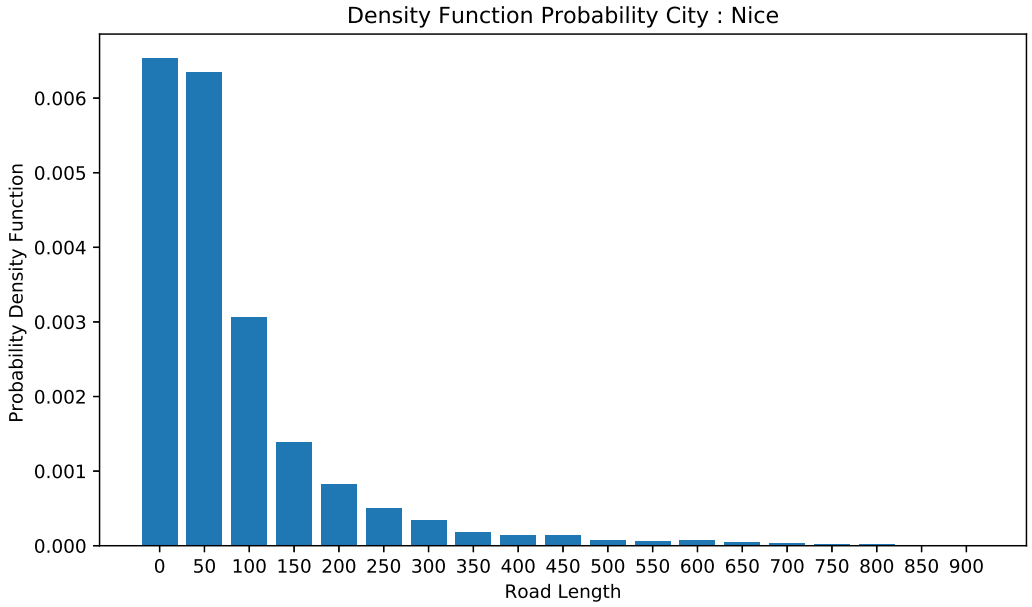

Figura 4.13: PDF di Nizza calcolata sul Length

La densità di probabilità delle angolazioni delle curve, rappresentata nella Figura 4.14, si differenzia dalle città europee per il suo andamento più irregolare. Questo si nota dalla presenza di picchi con frequenza incostante .

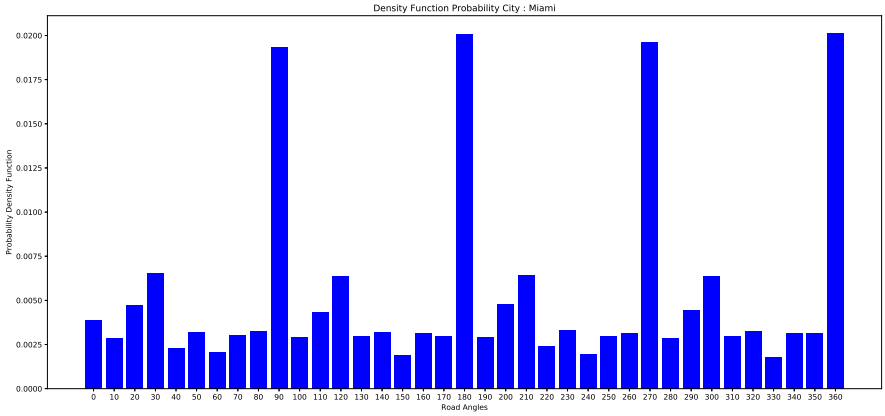

Figura 4.14: PDF di Nizza calcolata sul Bearing

4.4.1.4 Città di Milano

L'analisi della città di Milano viene riportata nella Figura 4.15, in cui viene rappresentata la PDF calcolata sulle lunghezze dei segmenti. Dal grafico compare in modo evidente che il numero di segmenti con maggiore probabilità di essere rinvenuti sono quelli da lunghezza da 0 a 50, da 50 a 100 le probabilità si dimezzano mentre le probabilità successive tenderanno verso 0.

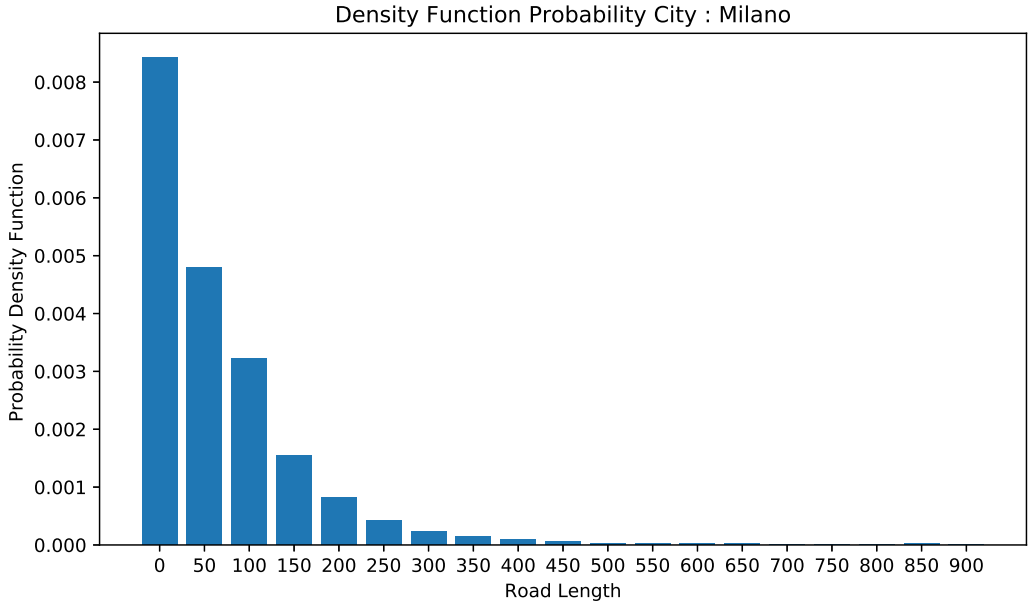

Figura 4.15: PDF di Milano calcolata sul Length

Nella figura 4.16 viene rappresentata la densità di probabilità delle angolazioni delle curve di Milano. Quest'ultime sembrano avere un andamento più regolare e presentano dei picchi il cui valore decresce costantemente.

Per avvalorare la tesi di quanto detto, sono state riportate graficamente dalla Figura 4.2 alla 4.5 le città prese in considerazione in ordine di descrizione. Si può notare infatti che le città descritte hanno una forma stradale molto simile tra loro, tranne per il fatto che il numero di nodi varia in base alla città. Questo spiega le poche differenze che intercorro nello studio dei grafici prima citati.

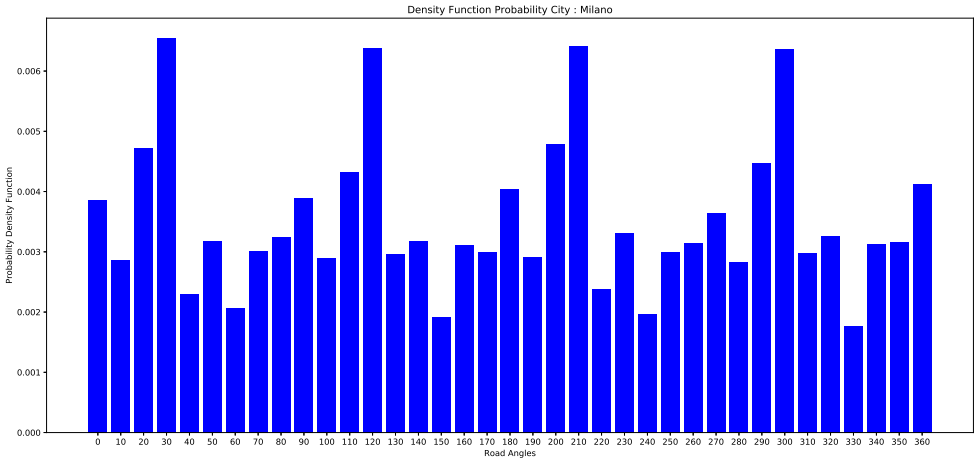

Figura 4.16: PDF di Milano calcolata sul Bearing

4.4.2 PDF delle Città Americane

Entriamo ora nel contesto Americano e tramite l'aiuto della PDF andremo ad analizzare le città di Miami e di Boston.

4.4.2.1 Città di Miami

Analizzando la città di Miami dalla Figura 4.19 si notano delle analogie con le città Europee: da 0 a 200 abbiamo una alta densità di probabilità(il picco maggiore si verifica a 50), dal 250 in poi i valori scendono drasticamente fino a da arrivare ad un andamento asintotico tendente a zero. D'altronde questo può essere spiegato semplicemente osservando che la tipica struttura di una città ammette segmenti stradali di quella lunghezza, specialmente se misurati tra uno svincolo e l'altro. Scenario comune è quello che, in prossimità del centro di una città, c'è più probabilità di riscontrare segmenti di strade corte a differenze di strade lunghe che compaiono raramente in una città con molti svincoli.

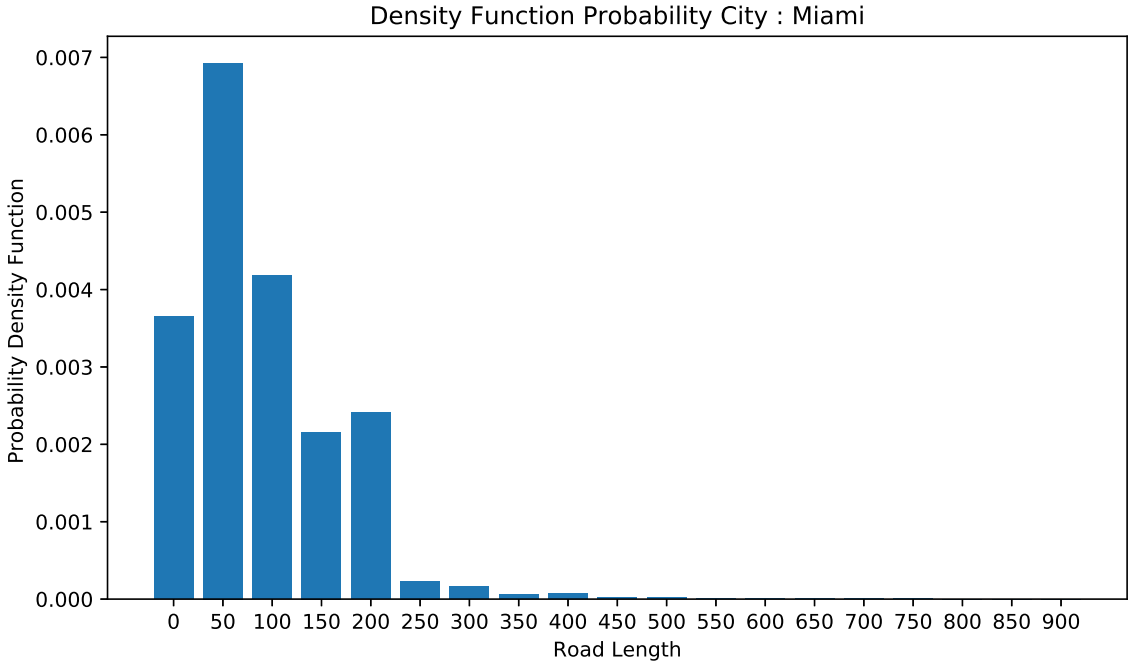

Figura 4.17: PDF di Miami calcolata sul Bearing

Nella Figura 4.20 viene evidenziata la classica struttura di un città americana, caratterizzata da un tipo di pianta a scacchiera, detta anche a griglia o a schema ippodameo, un tipo di pianta architettonica in cui le strade sono disposte in modo tale da formare angoli retti alla loro intersezione e una serie di isolati quadrangolari. Tale particolarità viene evidenziata nel grafico della figura da una struttura dove la probabilità di densità risulta molto bassa fatta eccezione per i quattro picchi presenti a 90, 180, 270 e 360.

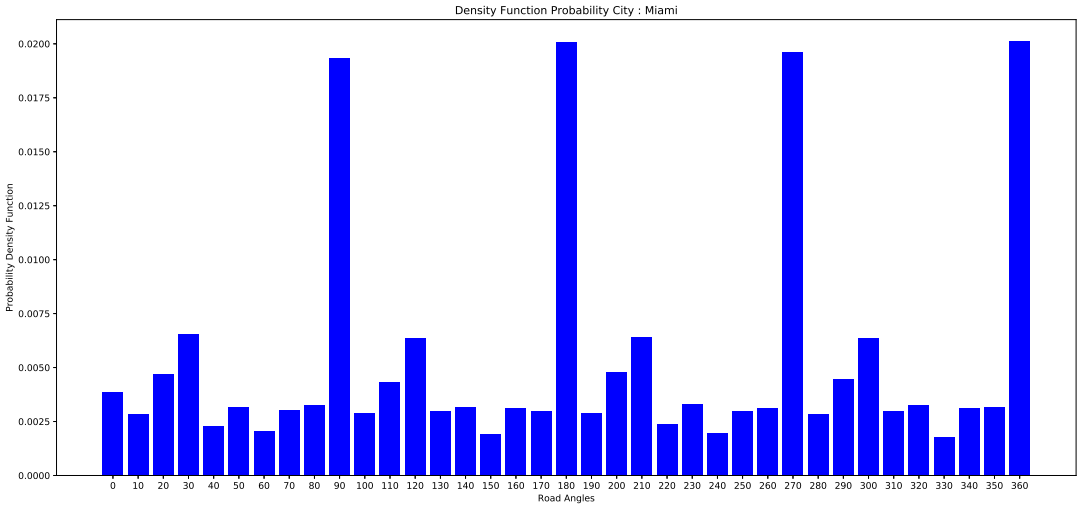

Figura 4.18: PDF di Miami calcolata sul Bearing

In questo contesto sono sorti alcuni problemi durante la ricerca dei percorsi come descritta nel Capitolo 3. In un contesto di città dove la forma risulta essere a scacchiera il numero di path trovate, anche nei casi in cui il rumore introdotto per la simulazione non fosse elevato, risulta essere maggiore rispetto alle città analizzate in precedenza come Bologna e Dublino.

4.4.2.2 Città di Manhattan

Nella Figura 4.17 viene raffigurata la densità di probabilità di Manhattan. Il grafico presenta un picco che evidenzia un abbondante presenza di segmenti di strade la cui lunghezza è approssimabile ad un intorno di 50. Il segmento 0 è paragonabile, per dimensione, al segmento di lunghezza 100; da 150 a seguire i valori decrescono fino a rasentare lo 0.

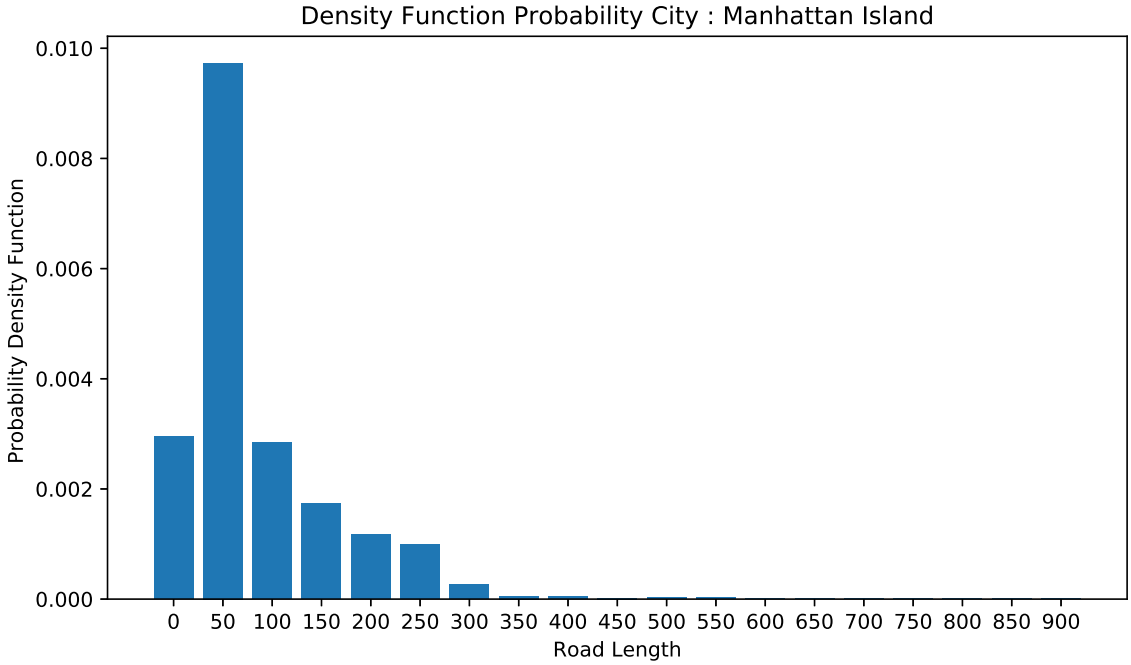

Figura 4.19: PDF di Manhattan calcolata sul Length

Per quanto riguarda la PDF delle inclinazioni delle curve la città Manhattan presenta la classica struttura delle metropoli americane: si possono notare dalla Figura 4.18 quattro picchi nelle angolazioni 0,90,180,270 riconducibili alla classica forma a scacchiera descritta anche per la città di Miami descritta precedentemente.

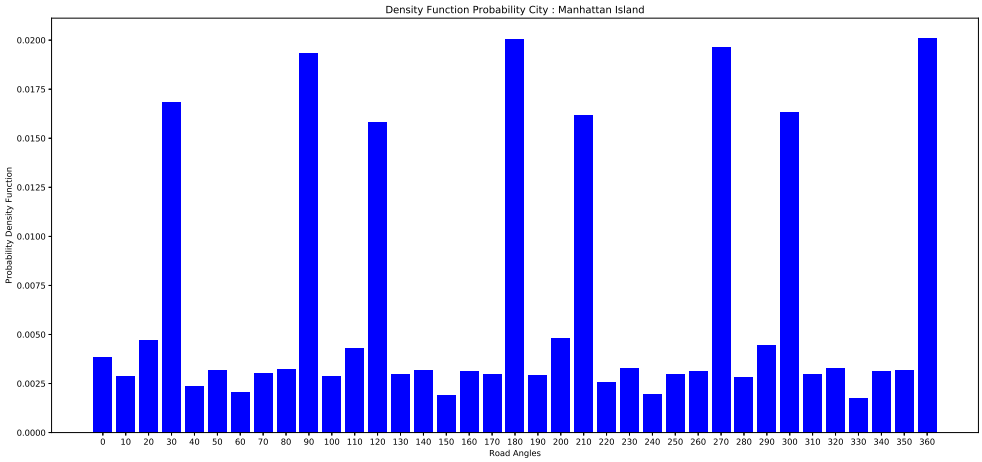

Figura 4.20: PDF di Manhattan calcolata sul Length

4.4.2.3 Città di Boston

Come ormai viene intuito, la PDF delle lunghezze dei segmenti (diversamente da quella delle angolazioni) tende ad essere molto simile per tutte le città. Questo risulta anche dalla Figura 4.21 dove viene riportata la PDF della città di Boston.

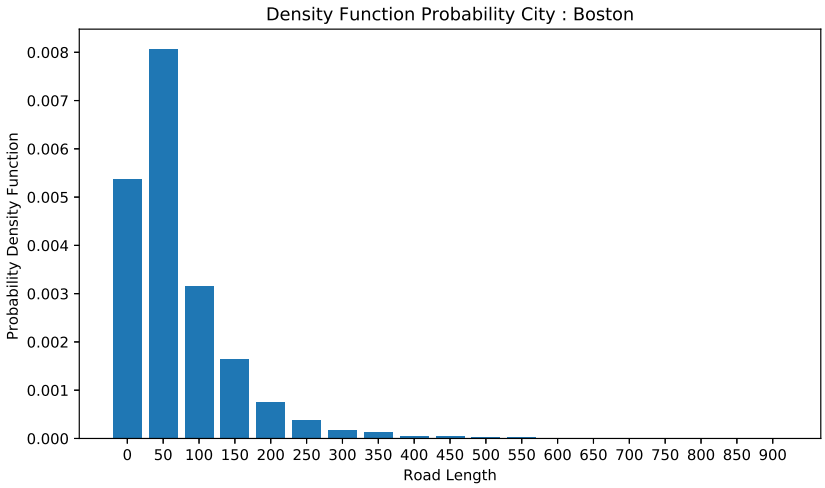

Figura 4.21: PDF di Boston calcolata sul Length

Per quanto riguarda la PDF delle angolazioni delle curve si può notare come, a differenza della città di Miami descritta in precedenza, Boston è una delle poche città americane che hanno una struttura simile alle città europee. Dalla PDF si può notare la somiglianza alle città europee.

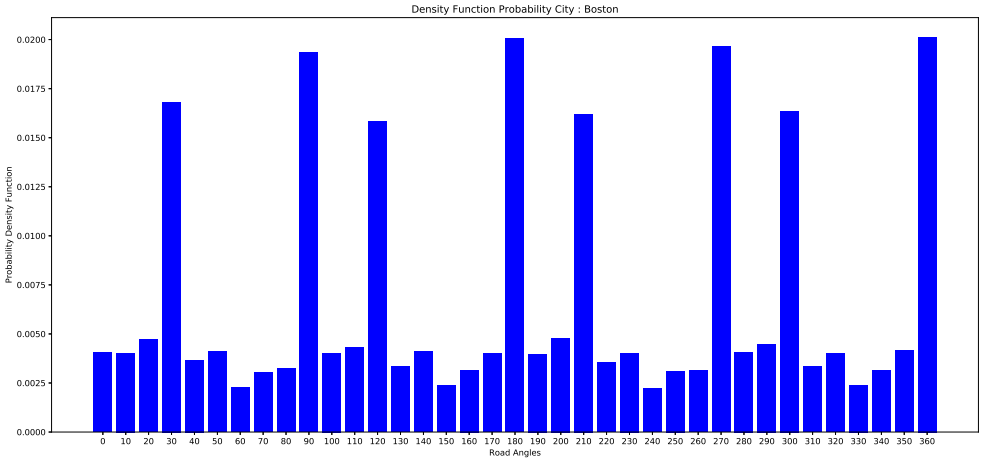

Figura 4.22: PDF di Boston calcolata sul Bearing

Viene descritto quanto detto in precedenza visivamente tramite le piante delle città riportate dalla Figura 4.6 alla Figura 4.8 che presentano differenze sostanziali: le piante di Miami e Manhattan risultano prevalentemente a scacchiera, il che comporta un aumento dei percorsi riconosciuti come percorsi validi specialmente su poche curve effettuate. Quindi in conclusione con una pianta di questo genere, un percorso più breve corrisponde ad avere meno probabilità di riuscita di match. In una pianta come quella di Boston, prevalentemente irregolare, anche se i percorsi ricercati sono brevi, sono maggiormente distinguibili fra loro, e l'algoritmo produce una lista più piccola di percorsi candidati.

4.5 Analisi dei percorsi con differenti rumori

In questa parte di analisi vengono richiamati i dati finali, prodotti dallo script "subGrapy" descritto precedentemente nel sotto paragrafo 3.5, dove si troverà una correlazione

con quanto illustrato nelle funzioni della PDF. L'analisi anche qui sarà effettua dividendo il contesto europeo da quello americano. Quindi sarà motivo di studio il numero di correlazioni trovate tra percorsi simulati e sotto grafi creati con l'aggiunta del rumore. Come detto in precedenza il rumore è una parte fondamentale per effettuare il match dei percorsi, può essere causato da varie motivazione come il sensore scalibrato oppure forte rumore elettrico prodotto della poco precisione dei sistemi inerziali presenti sugli smartphone. Il rumore preso in considerazione per questo studio viene classificato in :

- Rumore basso: che fa parte dalla soglia che va da 5 a 10.
- Rumore medio: che va dalla soglia 10 fino ad un massimo di 20.
- Rumore alto: che fa parte del range da 20 a 30.

4.5.1 Ansali grafica del rumore di tutte le città scelte

Per concludere, facendo un riassunto dell'analisi del rumore sui percorsi, nella Figura 3.18 vengono rappresentate graficamente tutte le città prese in considerazione per questo test, ad intervalli di rumore che variano tra 5, 10, 20 e 30, per capire al meglio le differenze che intercorrono tra i diversi tipi di città. E' subito evidente come le città di colore verde, verde chiaro, blu e viola riportano una media dei percorsi che non discosta molto da 1. Queste città sono proprio quelle la cui rete stradale non è rappresenta da una scacchiera, mentre per quanto riguarda le città riportate con il colore giallo e rosso il numero di percorsi travati cresce a dismisura quasi ad arrivare ad una media di 130 percorsi trovati.

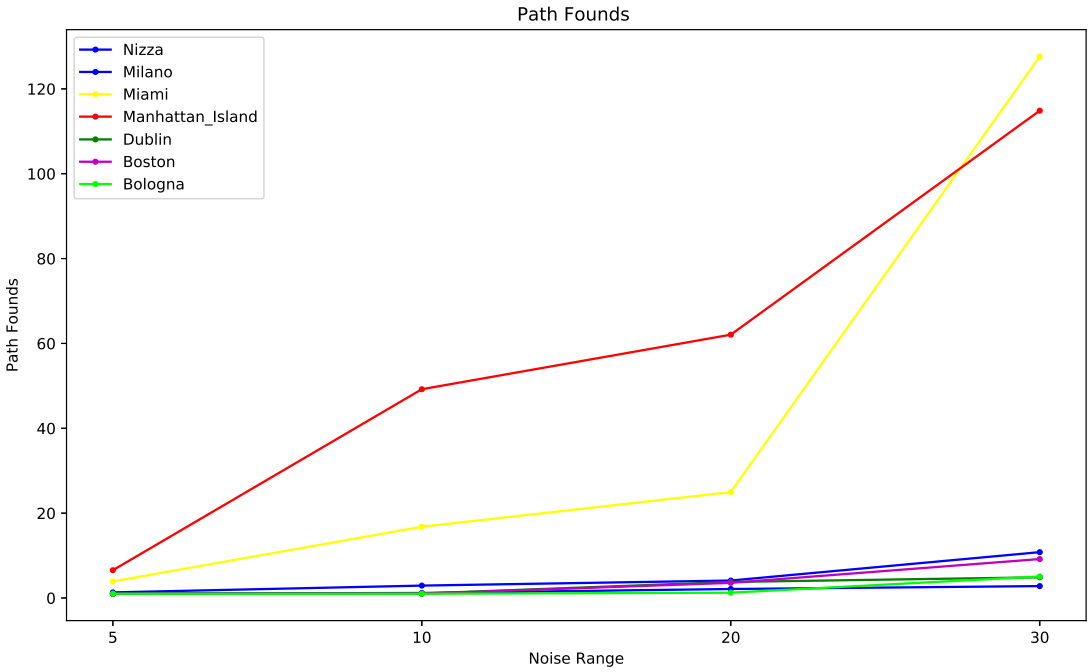

Figura 4.23: Analisi dei percorsi trovati al variare del rumore

4.6 Analisi dei percorsi con differenti rumori di una città Europea

In questa sezione si mette in evidenza la differenza che persiste tra le variazioni di rumore della città presa in considerazione. Nella Figura 4.24 viene analizzata la città di Bologna: nell'asse delle X vengono riportati il valore del rumore che va da 0 a 200 con intervalli regolari di 50. Nell'asse delle Y viene riportata la media del numero di percorsi trovati. L'obiettivo del grafico è quello di verificare il comportamento di una variabile al rimanere costante dell'altra. Viene analizzato ad esempio l'andamento della curva verde(Bearing): rimanendo costante il valore della Length si nota come aumenti la media dei percorsi trovati fino ad arrivare ad un valore massimo di 2.0 con un rumore di 150. La seconda curva da analizzare viene riportata con il colore rosso(Length): invariando il valore del

bearing si evince che la media dei percorsi trovati è minore rispetto alla curva descritta precedentemente, infatti i valori vanno da 1.0 ad un massimo di 1.2.

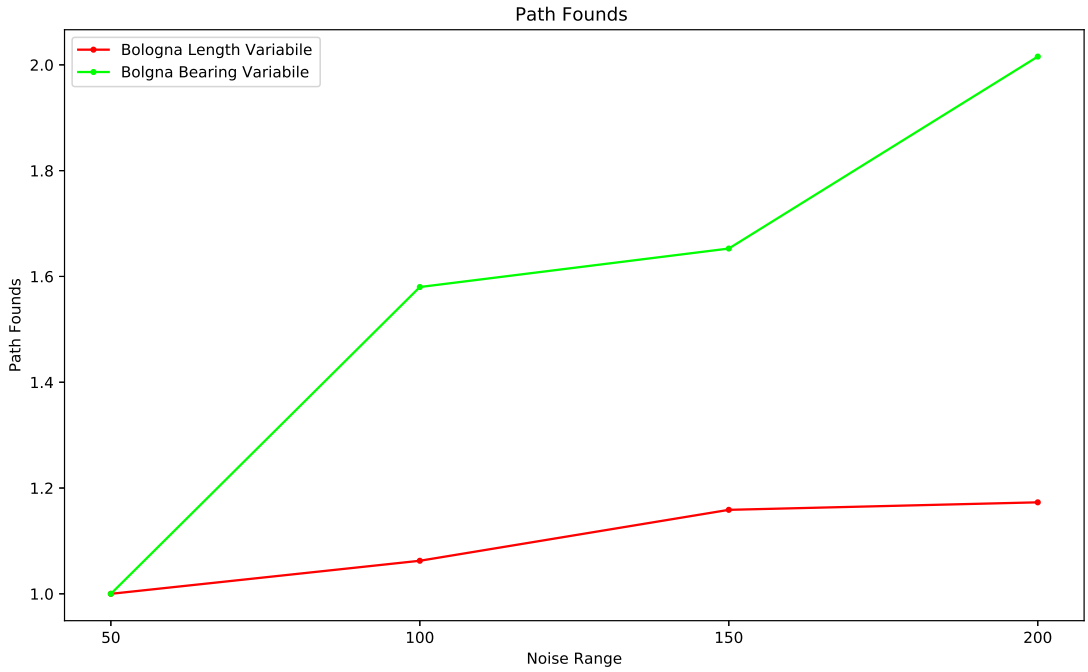

Figura 4.24: Verifica del rumore città di Bologna

4.7 Analisi dei percorsi con differenti rumori di una città Americana

In questa sezione analizziamo la differenza che intercorre tra le variazioni di rumore della città presa in considerazione. Nella Figura 4.25 viene analizzata la città di Santa Monica situata in California: nell'asse delle X vengono riportati il valore del rumore che va da 0 a 200 con intervalli regolari di 50. Nell'asse delle Y viene riportata la media del numero di percorsi trovati.

Viene analizzato ad esempio l'andamento della curva del Bearing riportata con il colore verde: rimanendo costante il valore della Length si nota come aumenti lentamente la

media dei percorsi trovati fino ad arrivare ad un valore massimo di 40. La seconda curva da analizzare viene riportata con il colore rosso(Length): tenendo costante il valore del bearing si nota che la media dei percorsi trovati è maggiore rispetto alla curva descritta precedentemente, infatti i valori vanno da un minimo di 20 ad un massimo di 140. Come già sottolineato più volte la differenza di valori ottenuti deriva dalla diversa struttura delle città americane rispetto a quelle europee.

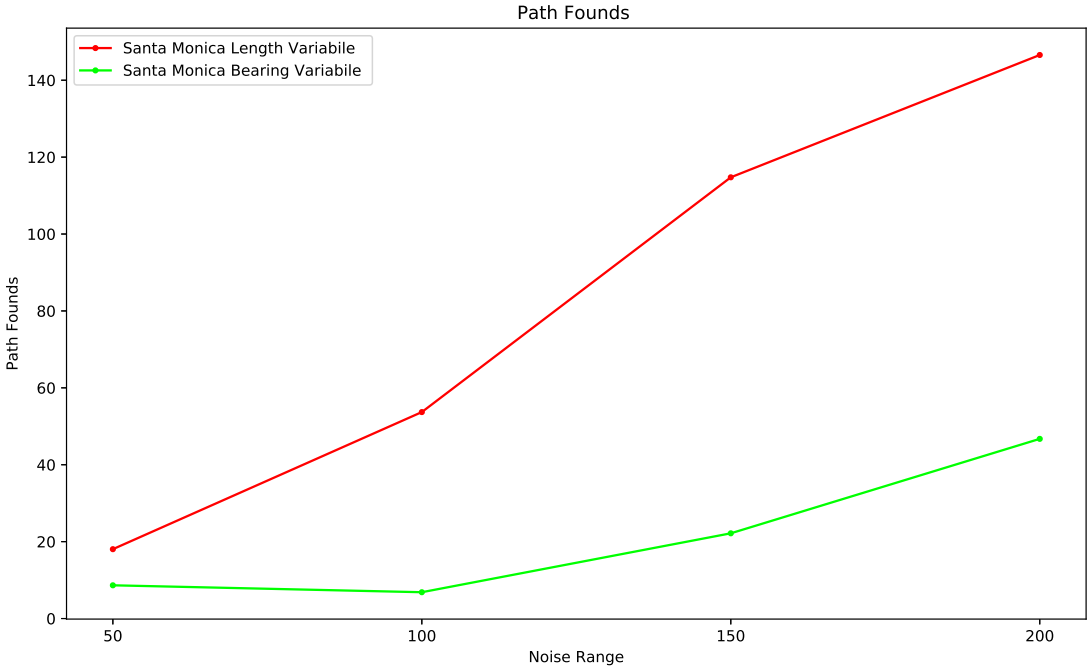

Figura 4.25: Verifica del rumore della città di Santa Monica, California

Capitolo 5

Conclusioni e sviluppi futuri

In questa tesi di laurea è stato sviluppato uno script Python che grazie alla creazione dei sotto grafi di una città cerca di recuperare un percorso effettuato dall'utente tramite la trasmissione dei dati dei sensori inerziali. Questo lavoro, oltre ad arricchire un campo non del tutto sviluppato, cerca di proporsi come applicazione innovativa nel settore, in quanto non utilizza il GPS per la ricerca dei percorsi bensì cerca di fare affidamento esclusivamente sui sensori inerziali, presenti nei moderni smartphone, minimizzando il consumo d'energia della batteria durante la sessione di guida.

Per rendere la simulazione maggiormente vicina ad un'applicazione reale, sono stati introdotti 4 ordini di rumore per emulare il comportamento reale di varie categorie di dispositivi, da quelli più precisi a quelli più grossolani. Viene considerato il rumore più basso per gli smartphone di ultima generazione fino ad arrivare ad un rumore più alto prodotto dai dispositivi più datati. Nello specifico, i risultati di questo studio mostrano un riconoscimento dei percorsi maggiori per le città Europee poiché, come descritto precedentemente, risultano avere un'architettura stradale più irregolare che facilita l'individuazione dei percorsi. Questo tipo di architettura consente un facile riconoscimento attraverso l'uso del magnetometro, perché gli archi presenti per ogni nodo assumono valori sufficientemente diversi, garantendo una maggiore singolarità dei percorsi.

Se invece si discutono i risultati ottenuti dall'analisi delle città Americane, anche con

dispositivi più precisi si riscontra molta più difficoltà nel trovare dei percorsi unici, poiché la maggioranza dei percorsi tendono ad avere una struttura di grafo simile,rendendo di conseguenza molto difficile il riconoscimento; questo avviene perché molti incroci (nodi) della mappa avranno archi ovvero strade con un valore di "Bearing" multiplo di 90 (0, 90, 180, 270). Non tutte le città americane presentano queste caratteristiche, basta vedere la città di Boston analizzata in precedenza la quale può essere paragonata ad un'architettura tipicamente europea. Resta particolarmente positivo il caso europeo, che detiene valori medi di percorsi trovati tendenti a risultati poco superiore a 1 in media, sia applicando un rumore basso che un rumore più alto. In aggiunta a quanto detto prima, viene considerato un altro fattore che determina la riuscita con ottimi risultati di questo esperimento: ovvero la lunghezza dei segmenti. Se i segmenti risultano molto corti allora le possibilità che si possano trovare dei percorsi simili, sopratutto con un alto rumore, sono maggiori. Quindi riassumendo, più è elevato la lunghezza media dei segmenti del percorso da trovare e minore sarà la probabilità di trovare corrispondenze multiple, con un'alta probabilità di ottenere un risultato unico.

Possibili sviluppi futuri potrebbero riguardare la creazione di un algoritmo che cerchi di contenere i costi computazionali della creazione dei sotto grafi in modo da avere un algoritmo più efficiente. Allo stesso modo si può cercare di velocizzare la ricerca ed il confronto tra i dati che vengono presi come input (dati simulati) ed i dati di tutti i sotto grafi. Per quanto riguarda la precisione dei percorsi trovati sarebbe opportuno cerca di integrare qualche altro sensore che non richieda i permessi per la lettura dei dati, per cercare di portare dei dati di ingresso più precisi rispetto a quelli adottati in questo elaborato, in modo tale da provare a ridurre il numero di percorsi trovati anche su città Americane. Un dato che potrebbe aiutare a migliorare queste prestazioni potrebbe essere un'analisi statistica sul tempo di esecuzione di un percorso.

Elenco delle figure

3.1 Architettura del sistema 17
3.2 Grafo della città di Modena 19
3.3 Porzione di grafo prima dell'unione dei nodi 24
3.4 Porzione di grafo unita 24
3.5 Riassunto grafico 25
3.6 Riassunto grafico 27
3.7 Grafico rappresentante la città di Miami 29
3.8 Grafico rappresentante la città di Milano 30
3.9 Schema completo dei procedimenti effettuati 30

4.1 Grafico dei nodi e degli archi delle città 33
4.2 Mappa Bologna 34
4.3 Mappa Dublino 34
4.4 Mappa Nizza 35
4.5 Mappa Milano 35
4.6 Mappa Miami 35
4.7 Mappa Manhattan 35
4.8 Mappa Boston 36
4.9 PDF di Bologna calcolata su "Length" 37
4.10 PDF di Bologna calcolata sulle Bearing 38
4.11 PDF di Dublin calcolata sul Length 39
4.12 PDF di Dublin calcolata sul Bearing 39
4.13 PDF di Nizza calcolata sul Length 40
4.14 PDF di Nizza calcolata sul Bearing 41

4.15 PDF di Milano calcolata sul Length 42
4.16 PDF di Milano calcolata sul Bearing 43
4.17 PDF di Miami calcolata sul Bearing 44
4.18 PDF di Miami calcolata sul Bearing 45
4.19 PDF di Manhattan calcolata sul Length 46
4.20 PDF di Manhattan calcolata sul Length 47
4.21 PDF di Boston calcolata sul Length 47
4.22 PDF di Boston calcolata sul Bearing 48
4.23 Analisi dei percorsi trovati al variare del rumore 50
4.24 Verifica del rumore città di Bologna 51
4.25 Verifica del rumore della città di Santa Monica, California 52

Listings

3.1 Codice per il recupero delle lunghezze 20
3.2 Recupero Informazioni inerenti alla Length 20
3.3 Funzione Chiamante 21
3.4 Funzione chiamata 22
3.5 Calcolo soluzione finale 26
3.6 Raffinare i risultati 28

Bibliografia

[1] Philippe Golle and Kurt Partridge. “On the anonymity of home/worklocation pairs”. In Hideyuki Tokuda, Michael Beigl, Adrian Friday,A. J. Bernheim Brush, and Yoshito Tobe, editors, Pervasive Computing, pages 390–397, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[2] A.R. Jimenez, F. Seco, C. Prieto, and J. Guevara. A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU. In 2009 IEEE International Symposium on Intelligent Signal Processing, pages 37–42. IEEE, aug 2009.

[3] Y. Michalevsky, A. Schulman, G. A. Veerapandian, D. Boneh, and G. Nakibly, “Powerspy: Location tracking using mobile device power analysis,” in Proceedings of the 24th USENIX Conference on Security Symposium. Washington, D.C.: USENIX Association, Aug. 2015, pp. 785–800.

[4] Jihe Wang, Meikang Qiu, Bing Guo, Yan Shen, Qiang Li. “Low-Power Sensor Polling for Context-Aware Services on Smartphones”, published in 2015 IEEE 17th International Conference on High Performance Computing and Communications

[5] A. Küpper, G. Treu, and C. Linnhoff-Popien, ”TraX: A device-centric middleware framework for location-based services, ” IEEE Communications Magazine, vol. 44, no. 9, pp. 114-120, Sept. 2006.

[6] FTC, “Android flashlight app developer settles FTC charges it deceived consumers,” https://www.ftc.gov/news-events/press-releases/2013/12/android-flashlight-app-developer-settles-ftc-charges-it-deceived, December 2013, accessed: November, 2015.

[7]] H. Leppakoski, J. Collin, and J. Takala. “Pedestrian Navigation Based on Inertial Sensors, Indoor Map, and WLAN Signals”. Journal of Signal Processing Systems, 71(3):287–296, jun 2013.

[8] Luca Bedogni, Marco Di Felice, and Luciano Bononi. “Context-aware Android applications through transportation mode detection techniques”. Wireless Communications and Mobile Computing, 16(16):2523–2541, nov 2016.

[9] S. Nawaz and C. Mascolo, “Mining users’ signicant driving routes with low-power sensors,” in Proceedings of the 12th ACM Conference on Embedded Network Sensor Systems, ser. SenSys ’14. ACM, 2014, pp. 236– 250.

[10] Vehicular Route Identication Using Mobile Devices Integrated Sensors Luca Bedogni, Luciano Bononi Department of Computer Science and Engineering, University of Bologna, Italy

[11] Pierluigi Casale, Oriol Pujol, and Petia Radeva. “Human activity recognition from accelerometer data using a wearable device”. Pattern Recognition and Image Analysis, 6669 LNCS:289–296, 2011.

[12] Sasank Reddy, Min Mun, Jeff Burke, Deborah Estrin, Mark Hansen, and Mani Srivastava. “Using mobile phones to determine transportation modes”. ACM Transactions on Sensor Networks, 6(2):1–27, 2010.

[13] A.R. Jimenez, F. Seco, C. Prieto, and J. Guevara. “A comparison of Pedestrian Dead-Reckoning algorithms using a low-cost MEMS IMU”. In 2009 IEEE International Symposium on Intelligent Signal Processing, pages 37–42. IEEE, aug 2009.

[14] Wonho Kang and Youngnam Han. “SmartPDR: Smartphone-Based Pedestrian Dead Reckoning for Indoor Localization”. IEEE Sensors Journal, 15(5):2906–2916, may 2015.

[15] OpenStreetMap, “OpenStreetMap Project,” https://www.openstreetmap.org/.

[16] S. Narain, T. D. Vo-Huu, K. Block, and G. Noubir. “The perils of user tracking using zero-permission mobile apps”. IEEE Security Privacy, 15(2):32–41, March 2017.

[17] Hongchao Fan, Alexander Zipf, Qing Fu, and Pascal Neis.” Quality assessment for building footprints data on OpenStreetMap”. International Journal of Geographical Information Science, 28(4):700–719, apr 2014.

[18] Python 3, https://www.python.org. 28(4):700–719, apr 2014.

[19] NetworkX, http://networkx.github.io/

[20] OSMnx,https://geoffboeing.com/2016/11/osmnx-python-street-networks/

[21] Algoritmo Dijkstra,E. W. (1959). ”A note on two problems in connexion with graphs” (PDF). Numerische Mathematik. 1: 269–271.

[22] Cormen, Thomas H.; Leiserson, Charles E.; Rivest, Ronald L.; Stein, Clifford (2001) [1990]. ”22.2 Breadth-first search”. Introduction to Algorithms (2nd ed.). MIT Press and McGraw-Hill. pp. 531–539. ISBN 0-262-03293-7.

[23] Matplotlib, https://matplotlib.org

[24] Marco Gruteser, Dirk Grunwald. ”Anonymous Usage of Location-Based Services Through Spatial and Temporal Cloaking”. Proceedings of MobiSys 2003: The First International Conference on Mobile Systems, Applications, and Services