

Report progetto di Geospatial Data Management

Simone Laudani

Scenario scelto

Tramite l'applicazione android: [GPSLogger](#), ho raccolto un insieme di punti relativi agli spostamenti durante una visita a Berlino.

Raccolta dati

I file sono in formato *gpx*, con la posizione espressa in gradi secondo il sistema di riferimento EPSG:4326 WGS 84 e timestamp in UTC con timezone. La frequenza di campionamento è stata impostata ad un minuto.

I dati sono stati raccolti durante un unico tragitto per ogni giornata, e secondo diversi metodi di spostamento: a piedi, bici e auto; senza soluzione di continuità nei cambiamenti di mezzo.

Ho registrato ogni cambiamento di mezzo manualmente dall'applicazione in tempo reale.

Obiettivo

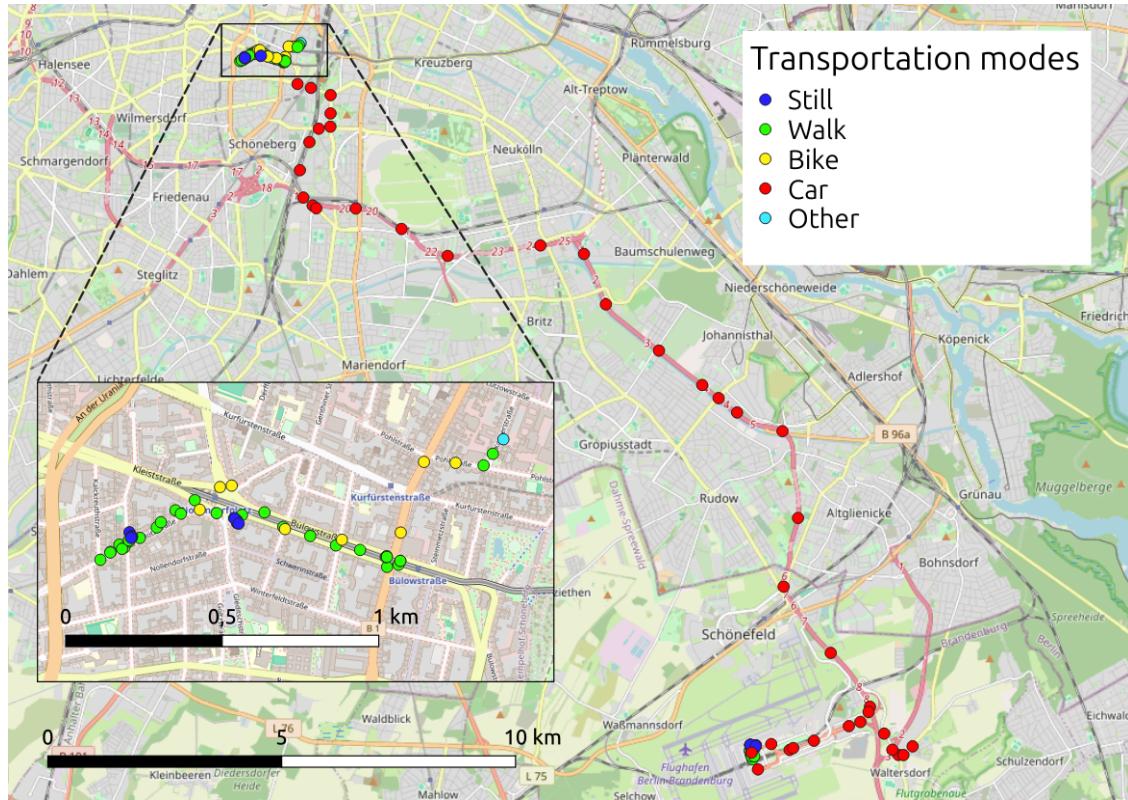
Come obiettivo ho cercato di predirre il metodo di spostamento per ogni punto, partendo dalle informazioni di posizione e timestamp.

Per verificare la bontà dei risultati ottenuti, ho usato la distanza tra predizione e *ground truth* come misura di errore:

$$\text{Errore} = \frac{\text{numero punti con predizione errata}}{\text{numero punti totali}}$$

E provato a ridurre il più possibile questo errore.

Preparazione dati



Mappa del percorso

Ottenuti i file gpx dall'applicazione, li ho caricati sul database Postgres attivando l'estensione PostGIS, su cui poter svolgere delle prime operazioni di preprocessing e pulizia.

Comando CLI usato per caricare gpx in Postgres:

```
$> ogr2ogr -f "PostgreSQL" PG:"host=localhost dbname=berlin user=xxx password=xxx" |  
  20250520.gpx |  
  -nln gps_20250520_original |  
  -overwrite
```

Qui gli attributi principali sono:

- **ogc_fid**: che ho trattato come id dei punti, dato che tutti i punti appartengono alla stessa traiettoria/segmento.
- **time**: il timestamp di creazione del dato, in UTC + timezone.
- **mode**: la modalità di trasporto, espressa come stringa a scelta tra: "still", "walk", "bike", "car".
- **src**: la sorgente di locazione, che può avere come valore "gps" o "network". All'aperto è usato il gps, mentre negli spazi chiusi o strade strette si fa un fallback sul network.
- **wkb_geometry**: il campo di geometria in gradi e da convertire per poterlo visualizzare e utilizzare per le query.

Ho omesso tutti gli altri campi non rilevanti, ad esempio i tracks o le altitudini.

Quindi ho creato una nuova tabella su cui poter svolgere le operazioni:

```
CREATE TABLE gps_20250520_processed AS  
(SELECT ogc_fid AS id, wkb_geometry, time, mode, src,  
     LAG(wkb_geometry) OVER (ORDER BY time) AS prev_geom,  
     LAG(time) OVER (ORDER BY time) AS prev_time  
  FROM gps_20250520_original  
 ORDER BY time);
```

```
ALTER TABLE gps_20250520_processed ADD PRIMARY KEY (id);  
CREATE INDEX ON gps_20250520_processed (id);
```

Aggiunto un nuovo attributo di tipo *geometry*, traducendo il formato di quello già presente:

```
ALTER TABLE gps_20250520_processed ADD COLUMN geom geometry(Point, 4326);
```

Popolo:

```
UPDATE gps_20250520_processed SET geom =  
ST_SetSRID(ST_GeomFromEWKB(wkb_geometry), 4326);
```

E aggiunto un nuovo campo *prev_geom*, per poter confrontare le coppie di punti:

```
UPDATE gps_20250520_processed SET prev_geom =  
ST_SetSRID(ST_GeomFromEWKB(prev_geom), 4326);
```

Ho aggiunto due nuovi attributi:

- **spatial_delta_m**: la distanza in metri tra il punto corrente ed il precedente.
- **temporal_delta_s**: il tempo trascorso in secondi tra il punto corrente ed il precedente.

```
ALTER TABLE gps_20250520_processed ADD COLUMN spatial_delta_m DOUBLE  
PRECISION;
```

```
ALTER TABLE gps_20250520_processed ADD COLUMN temporal_delta_s DOUBLE  
PRECISION;
```

```
UPDATE gps_20250520_processed SET spatial_delta_m = ST_Distance(geom::geography,  
prev_geom::geography);
```

```
UPDATE gps_20250520_processed SET temporal_delta_s = EXTRACT(EPOCH FROM (time -  
prev_time));
```

Ho dedotto la velocità in km/h per ogni punto secondo la semplice formula (la scelta di km/h è solo per una più facile intuizione rispetto a m/s):

$$\text{velocità [km/h]} = \frac{\Delta \text{spazio [m]}}{\Delta \text{tempo [s]}} * 3.6$$

```
ALTER TABLE gps_20250520_processed ADD COLUMN speed_km_h DOUBLE PRECISION;
```

```
UPDATE gps_20250520_processed SET speed_km_h = (spatial_delta_m / temporal_delta_s)  
* 3.6;
```

Infine ho preparato l'attributo per la predizione della modalità di trasporto:

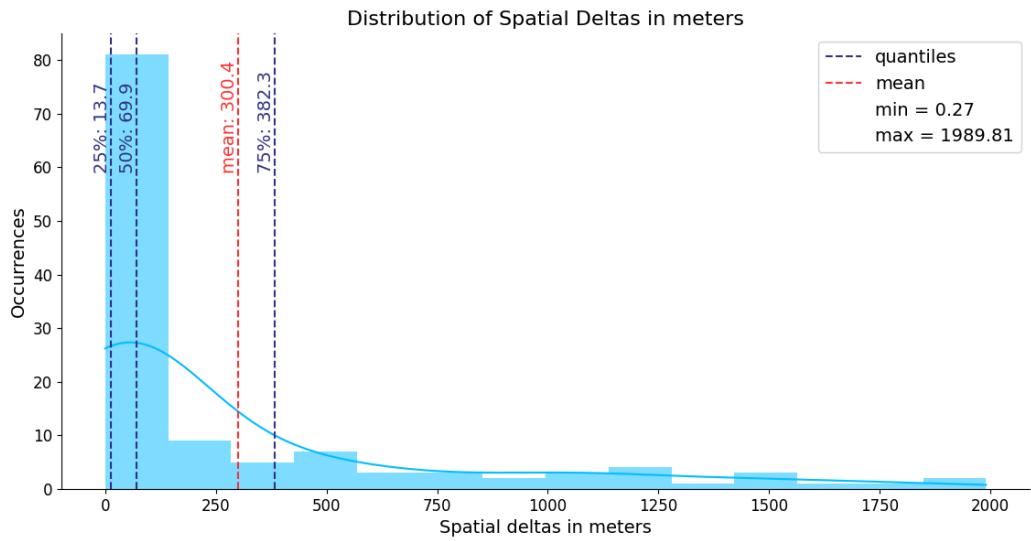
```
ALTER TABLE gps_20250520_processed ADD COLUMN inferred_mode_speed_const  
VARCHAR(16);
```

Statistiche dati

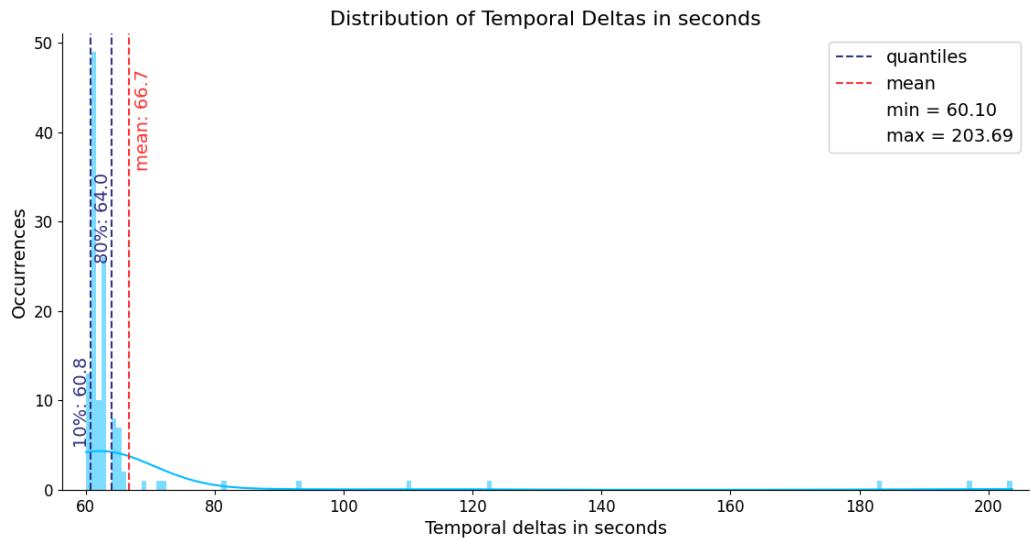
Eseguendo diverse query, si possono ottenere delle statistiche sui dati:

```
SELECT COUNT(*) as number_of_points, sum(spatial_delta_m) AS total_distance_m,  
MAX(time) - MIN(time) AS total_duration  
FROM gps_20250520_processed;
```

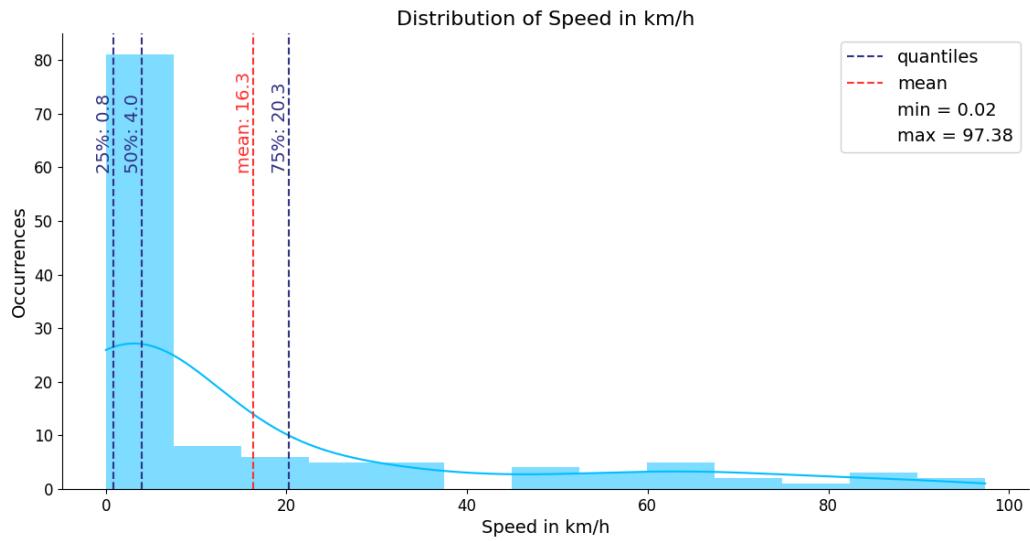
Base statistics	
Metric	Value
Number of points	126
Total distance (m)	37551.58813949
Total duration (HH:mm:ss)	2:19:02.141



Dalla distribuzione degli spostamenti, possiamo notare come la maggior parte di questi è al di sotto di 100m, mentre quelli più lunghi arrivano anche a 1.5 km.
La deviazione standard infatti è di 468.96.



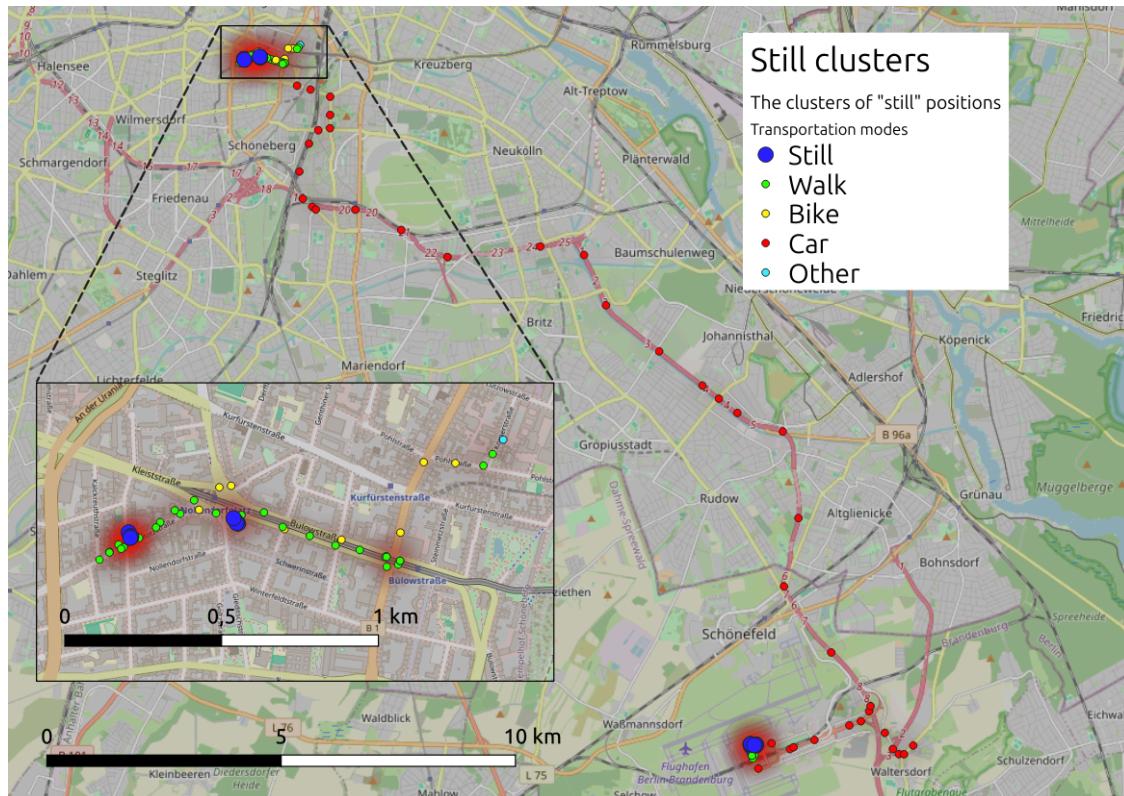
La distribuzione dei tempi invece è più omogenea rispetto a quella spaziale. Qui i tempi più lunghi indicano probabilmente una latenza nella cattura del segnale.



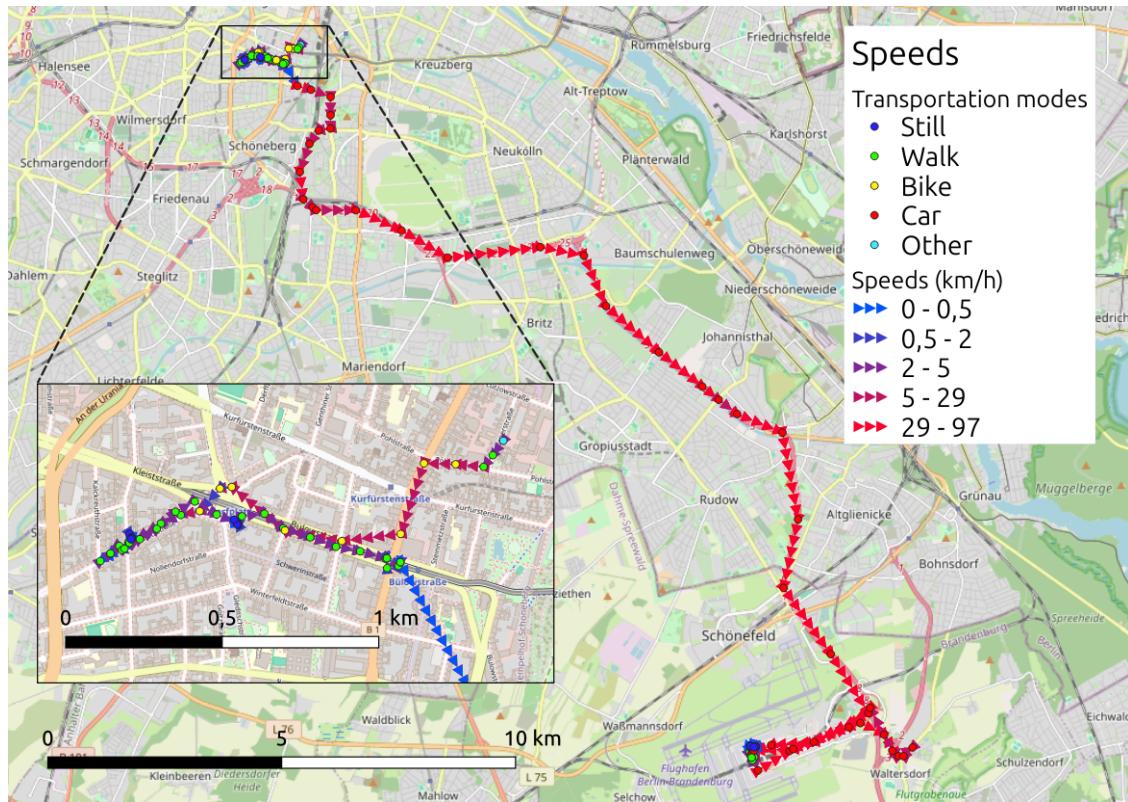
Le velocità vanno da valori vicino allo zero, mentre ero fermo, a quasi 100 km/h, non perchè correvo veloce, ma mi trovavo in auto in autostrada. Si notano gli spostamenti a velocità maggiori nella parte centrale della mappa geografica dei punti.

La maggior parte degli spostamenti sono comunque avvenuti a velocità molto contenute, quindi la maggior parte è avvenuta a piedi o bici.

Qui una mappa creata con DBSCAN, con numero minimo di punti nel cluster: 4, e distanza massima tra i punti di 30 metri, che mostra le aree che dovrebbero essere quelle in cui la modalità era "still" ovvero "fermo".



Possiamo notare che effettivamente la heatmap è ben sovrapposta ai punti "still" in blu.



Qui invece la visualizzazione delle velocità sui segmenti colleganti i punti.

Risoluzione e risultati

Per risolvere il problema di deduzione delle modalità di spostamento ho usato delle soglie di velocità con cui ottenere le 4 classi: **still, walk, bike, car**.

Per settare i valori ho usato la query:

```
UPDATE gps_20250520_processed
SET inferred_mode_speed_const =
CASE
    WHEN speed_km_h < threshold_1 THEN 'still'
    WHEN speed_km_h BETWEEN threshold_1 AND threshold_2 THEN 'walk'
    WHEN speed_km_h BETWEEN threshold_2 AND threshold_3 THEN 'bike'
    WHEN speed_km_h > threshold_3 THEN 'car'
    ELSE NULL
END;
```

Per capire quanto sono corretti i valori ottenuti, confronto il campo dedotto con quello della ground truth, mediante la query:

```
SELECT
    COUNT(*) FILTER (WHERE inferred_mode_speed_const IS DISTINCT FROM mode) AS
errors_number,
    COUNT(*) AS total_points,
    (COUNT(*) FILTER (WHERE inferred_mode_speed_const IS DISTINCT FROM mode) /
    COUNT(*)::FLOAT) AS error_ratio
FROM gps_20250520_processed
WHERE mode IS NOT NULL;
```

Con dei valori di soglie rispettivamente di: 2, 5 e 15 km/h impostati manualmente, ho ottenuto un tasso di errore di 0.176:

errors_number	total_points	error_ratio
22	125	0.176
(1 row)		

Ho quindi creato uno script python per trovare automaticamente attraverso una ricerca casuale, i valori ottimali delle soglie.

Consultabile in:

https://github.com/simone-05/gdm_project/blob/main/random_search_with_csv.py

Impostando un range per ogni soglia (in km/h):

- still-walk: [0, 5]
- walk-bike: [5, 10]
- bike-car: [10, 20]

Per ogni ciclo quindi si proveranno delle soglie casuali dentro il rispettivo range e confrontando otterremo il tasso di errore nuovo. Salvando le nuove soglie ogni volta che questo si abbassa.

In questo modo sono riuscito ad ottenere un errore minimo di:

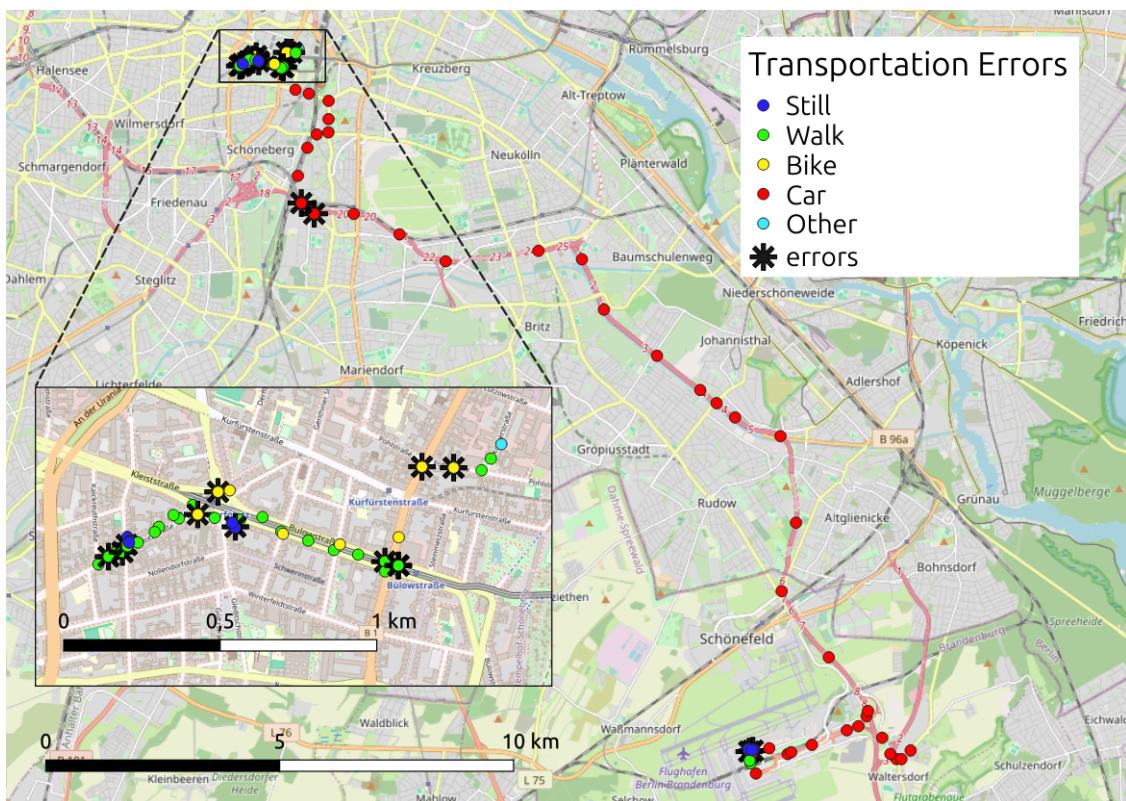
Errors statistics	
Metric	Value
Number of errors	15.0
Error ratio	0.12

Un dettaglio degli errori:

Error stats details

Metric	Value
Correctly predicted car	39 out of 41
Correctly predicted still	36 out of 39
Correctly predicted walk	31 out of 37
Correctly predicted bike	4 out of 8
Missed walk, predicted as still	6
Missed bike, predicted as walk	4
Missed still, predicted as walk	3
Missed car, predicted as walk	1
Missed car, predicted as bike	1

I punti in errore visualizzati sulla mappa:



I punti più difficili da predirre sono sicuramente quelli in bici, perchè ha caratteristiche a metà tra auto e piedi, ed è facilmente confondibile con entrambi.

Sia per velocità, un'auto in centro città mantiene velocità basse, comparabili con quelle di una bici.

Sia per posizione sulla strada, dato che può trovarsi sia carreggiata o marciapiede con pista ciclabile.

Riferimenti

GPSLogger: <https://gpslogger.app>