Task-01

Given a set of latitude and longitude coordinates, extract the corresponding pincode (postal code) for each location using OpenStreetMap (OSM) data.

OSM data is built from 3 core elements:

1. Nodes – single points (with lat/lon), e.g., a tree, lamp post
2. Ways – ordered list of nodes forming roads, areas, buildings
3. Relations – groupings of nodes/ways that define complex areas like cities or postal zones (e.g., multipolygon for a pincode)

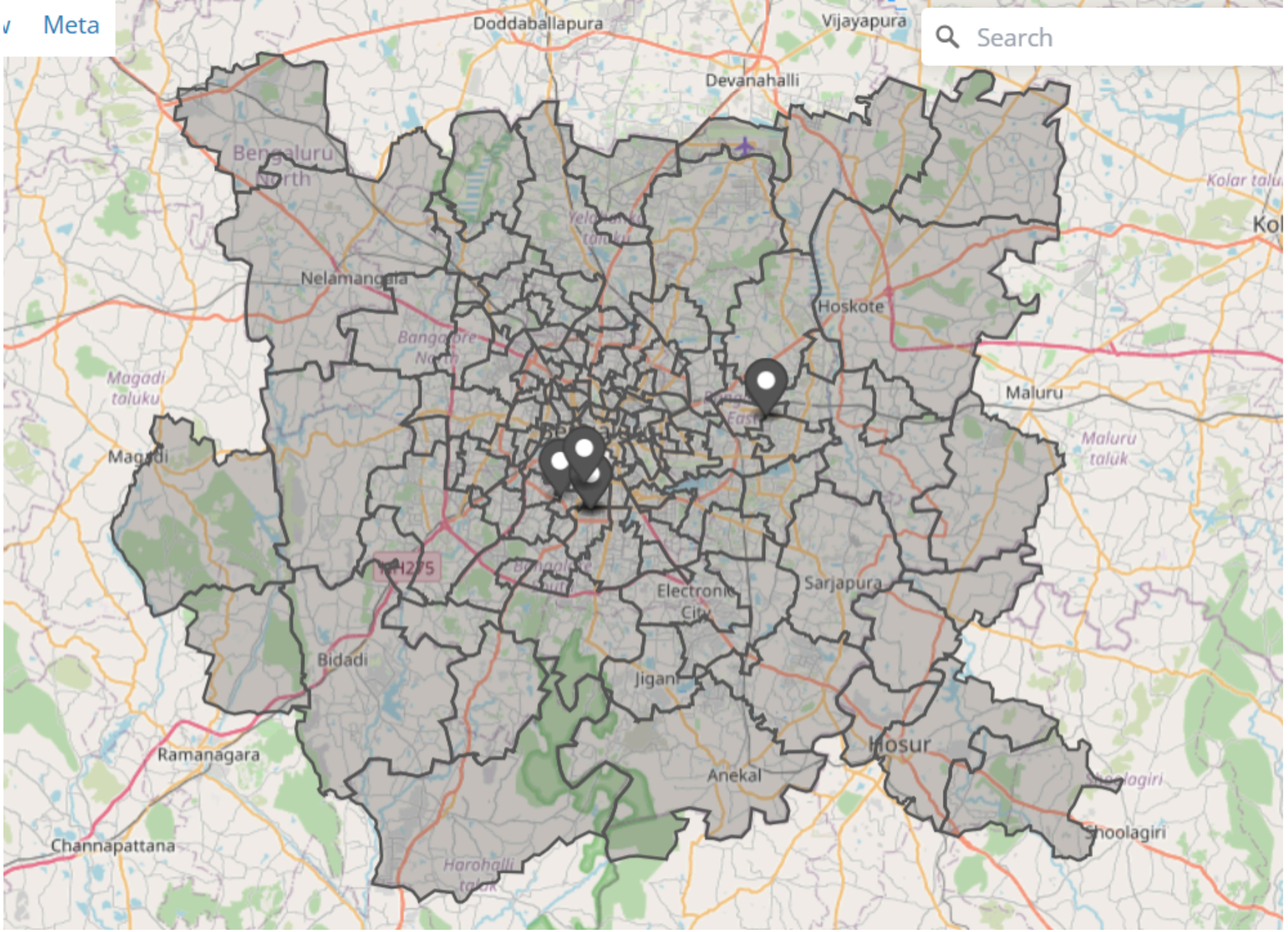
Osmium Tool:

Extract features with specific tags (e.g. all postal boundaries)

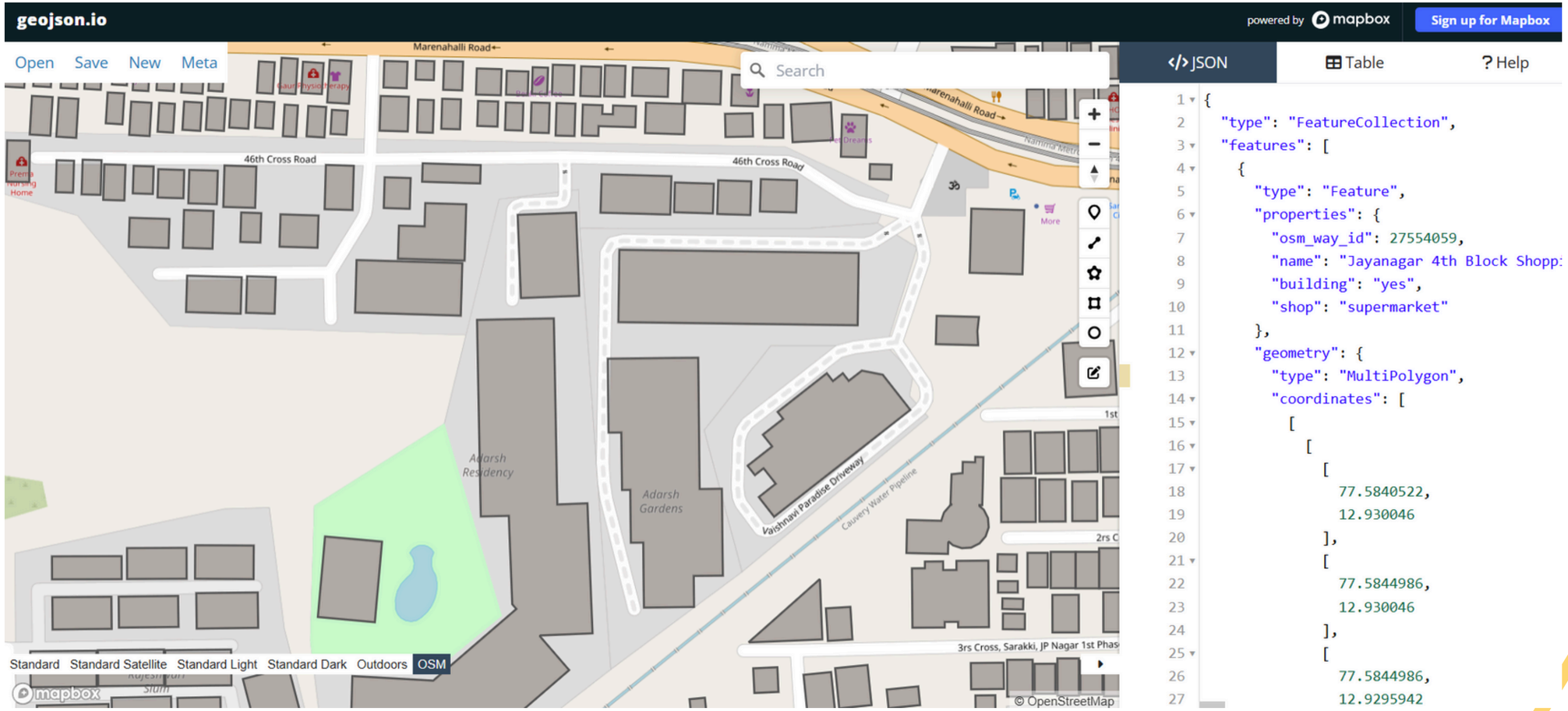
Stream large .pbf files efficiently

Export data to GeoJSON, CSV, or shapefiles

- Reverse geocoding
- Geofabrik


Meta
Search
Doddaballapura
Vijayapura
Devanahalli
Bengaluru North
Nelamangala
Hoskote
Magadi taluku
Magadi
Maluru
Maluru taluk
Sarjapura
Electronic City
Bidadi
Jigani
Hosur
Ramanagara
Anekal
Channapattana
Shoolagiri

"To handle cases where multiple delivery points exist within the same building, I utilize OSM way IDs associated with buildings. Since each building in OSM has a unique way ID, I group points by this ID to identify whether they belong to the same structure. This ensures that all deliveries within the same building are completed together before moving to a different building, avoiding unnecessary back-and-forth based purely on distance."



Problem Statement :

Detection of unwanted stoppages in vehicle trajectories using raw GPS data comprising latitude, longitude, speed, and timestamp information. The objective is to identify and flag unplanned or abnormal halts by analyzing spatio-temporal and speed characteristics, thereby enabling effective driver monitoring .

Raw GPS data is often noisy and inconsistent. Preprocessing involves:

1. Data Cleaning:
   Remove clearly wrong or inconsistent GPS points(GPS jumps).
2. Data Filtering & Smoothing:
   Reduce GPS "jitter" — tiny unwanted fluctuations even when the vehicle is stationary.

I explored several research papers that helped me understand how to clean GPS data effectively and detect meaningful stop behavior.

1.A Data Cleaning Method for Big Trace Data Using Movement Consistency

How It Works:

Step 1: Trajectory Segmentation

- The raw GPS trajectory is split into segments based on motion changes.
- This uses position interference (verdisk) and angle jamming (angdisk).

- verdisk: If a point strays too far from the straight line between its neighbors, it's a sign of inconsistency.
- angdisk: If the angle between consecutive path segments is too sharp, that point is considered a potential split point.
- Thresholds a1 (angle) and a2 (distance) are used to decide whether a point causes a new segment.

Step 2: Movement Consistency Model

For each sub-trajectory:

- Apply RANSAC (Random Sample Consensus):
  1. Try fitting a straight line to the segment.
  2. Points that deviate from this line beyond a similarity threshold are marked as noise or outliers.

Good for:

It is well-suited for noisy or messy GPS data where the goal is to clean up large inconsistencies in motion.

2.Simplifying GPS Trajectory Data with Enhanced Spatial-Temporal Constraints (ESTC-EDP)

Douglas–Peucker reduce the number of points by checking only the shape of the path. That is:

- If a point is close to the straight line between two others, it gets removed.

But in real-world applications (like detecting stops, turns, or speed changes), these simple points might actually be very important behavior indicators.

What ESTC-EDP Does Better

This paper proposes an improved algorithm that keeps points not just based on shape — but also based on behavior.

It adds four enhanced constraints (the “ESTC” part):

- Speed constraint: retains fast/slow transitions
- Time constraint: retains stops based on time
- Spatial constraint: retains behavior zones (road, parking, etc.)
- Elevation constraint: retains altitude changes (flyovers)

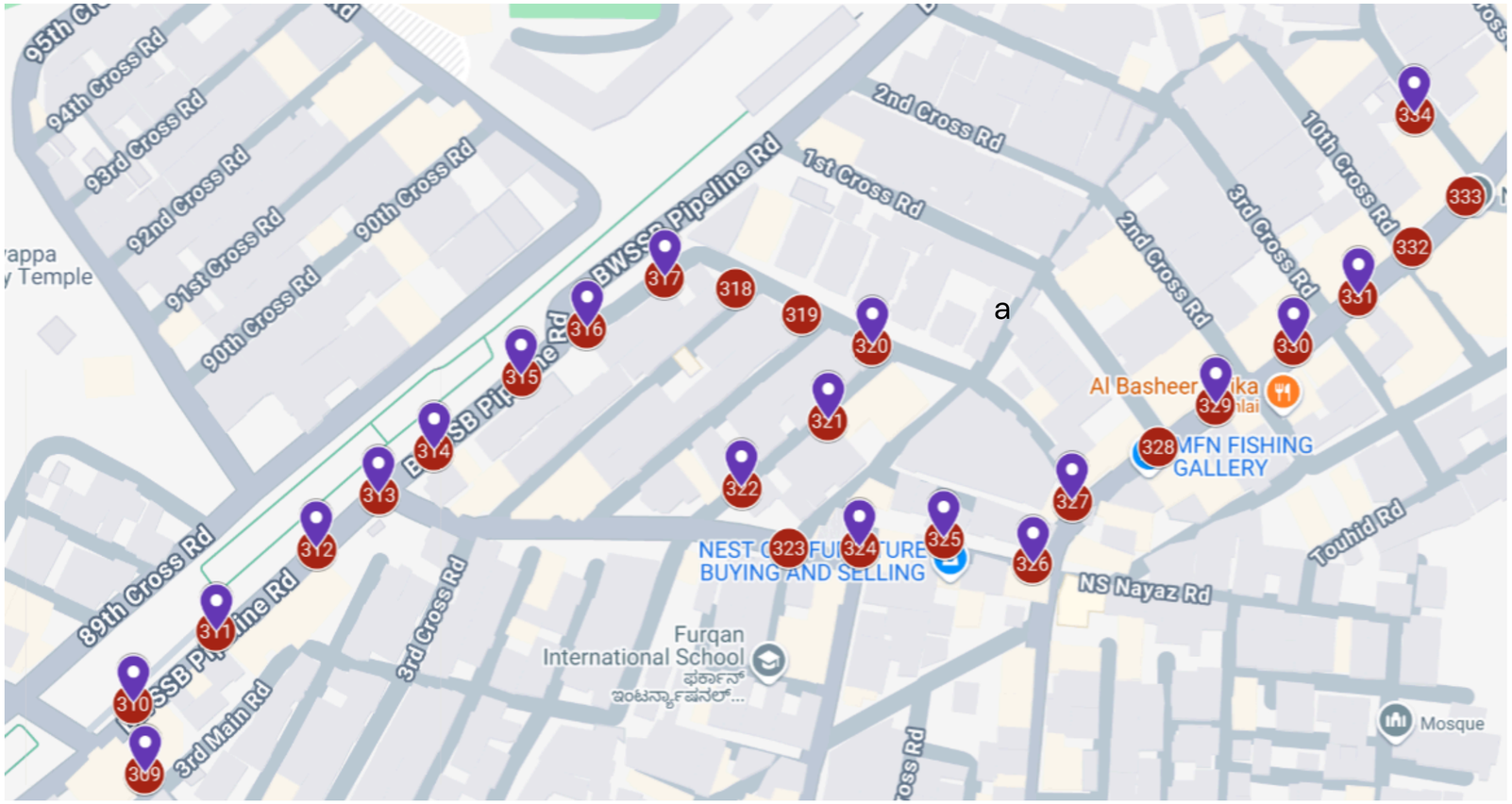
How it works?

- Each of the four constraints uses a threshold.
- If the point is normal and within the threshold range, it's not important ,so it’s removed.
- But if the value crosses the threshold, it means something meaningful happened — like a stop or speed change ,so that point is kept.”
- ESTC-EDP is applied before map matching.
- Although it uses haversine distance and might preserve off-road points, that’s acceptable because the goal is to retain behavioral signals like stops or speed changes.
- Then, map matching aligns those cleaned points back to the correct road geometry.

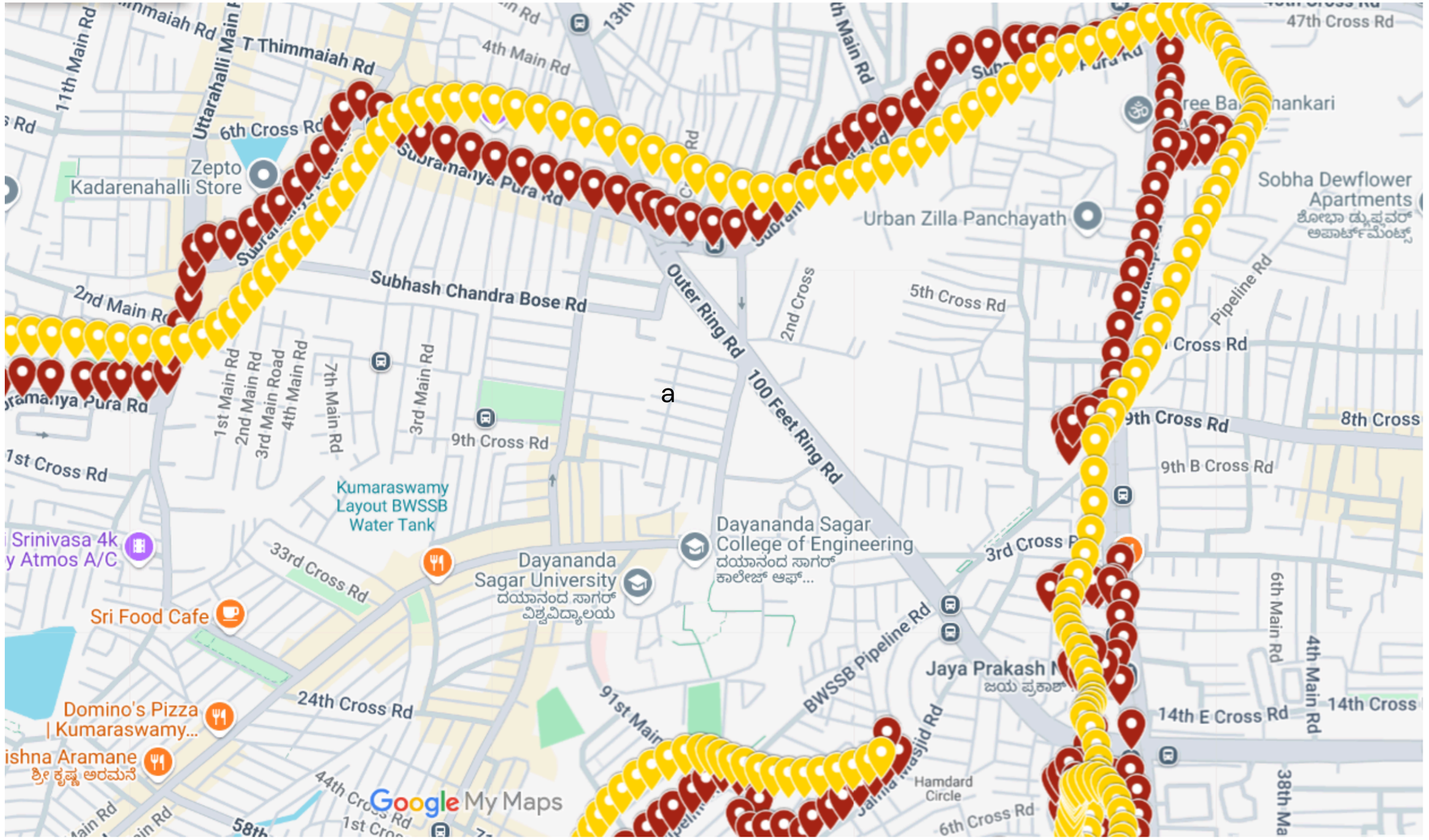
Advantages:

1.It removes points with no behavioral variation, making the data cleaner, more focused, and ensuring faster and more reliable stop detection.

2. Preserves Critical points



kalman filter



Valhalla Map Matching

What is Map Matching?

Map matching is the process of aligning raw GPS points (which can be noisy or slightly off) to the actual roads on a map.

What is Valhalla?

Valhalla is an open-source routing and map-matching engine developed by Mapbox.

a

It provides tools for:

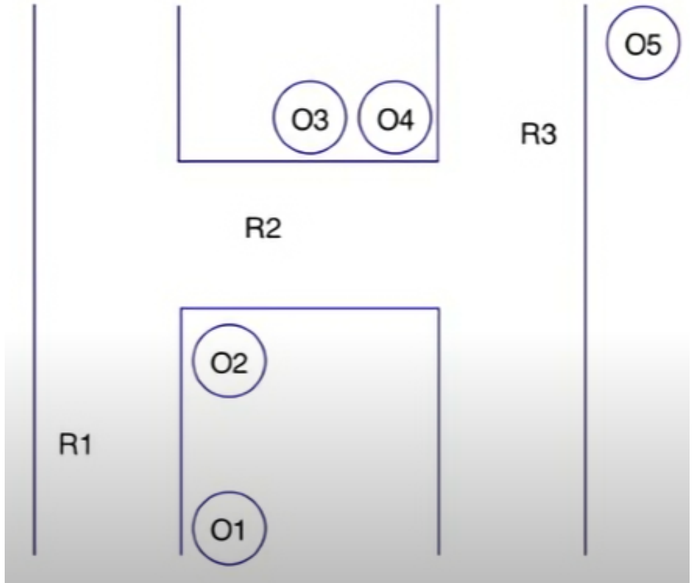
- Turn-by-turn navigation
- Route planning
- Isochrone and matrix calculations
- Map matching

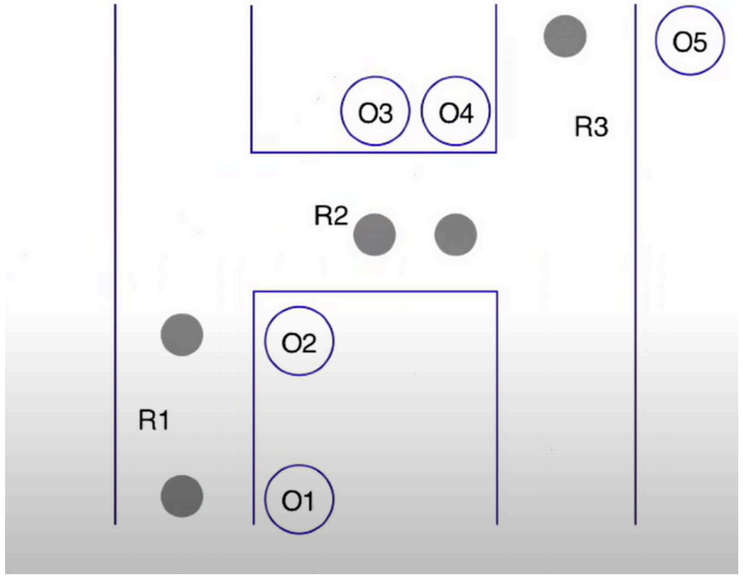
Why Valhalla Map Matching is Good?

- Open-source & free
- Highly customizable
- Self-hostable with Docker
- Parallel processing support
- uses spatial, temporal, and speed context along with nearby trace behavior

How valhalla works?

Valhalla’s Meili module performs map matching using a Hidden Markov Model (HMM) combined with the Viterbi algorithm.

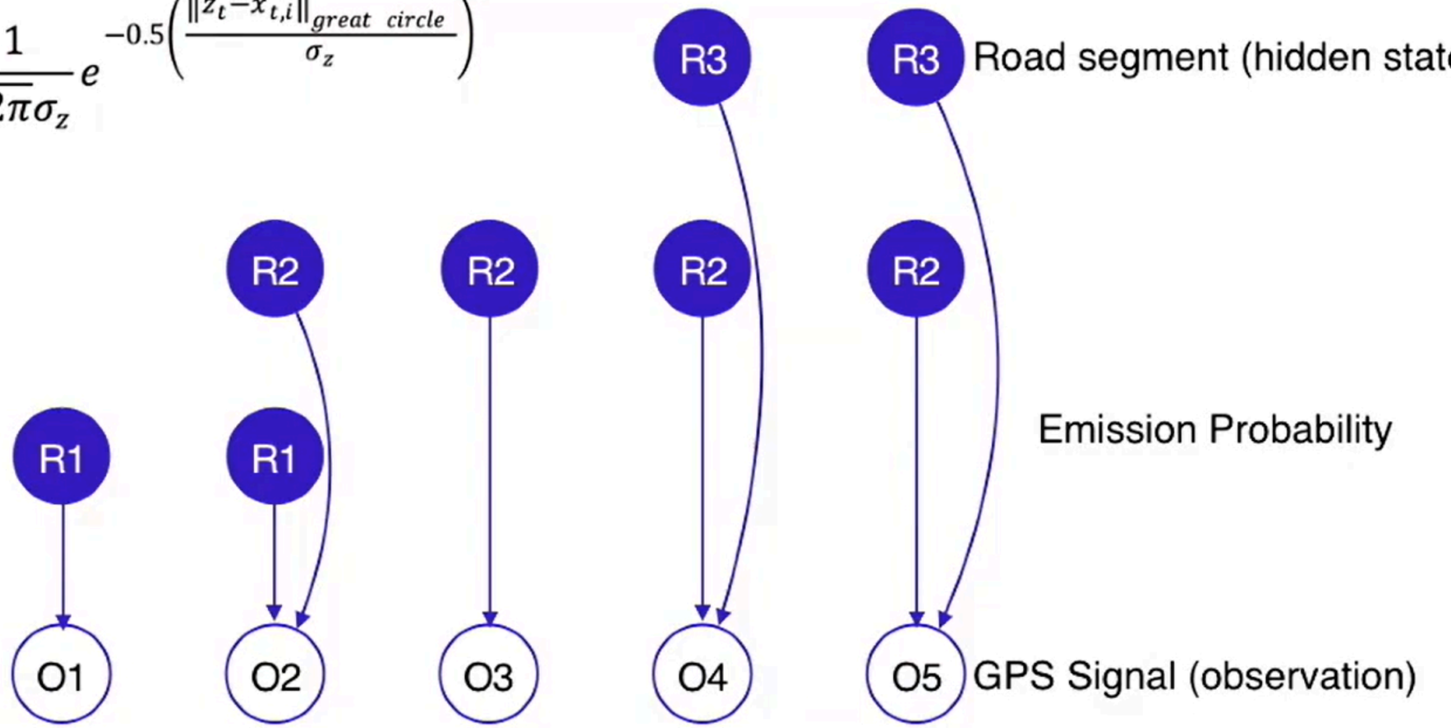




Valhalla uses Hidden Markov Model (HMM) where:

- Hidden states = Actual road segments (e.g., R1, R2, R3...)
- Observations = Noisy GPS points (O1, O2, O3...)

$$p(z_t|r_i) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{-0.5\left(\frac{\|z_t - x_{t,i}\|_{great\ circle}}{\sigma_z}\right)^2}$$



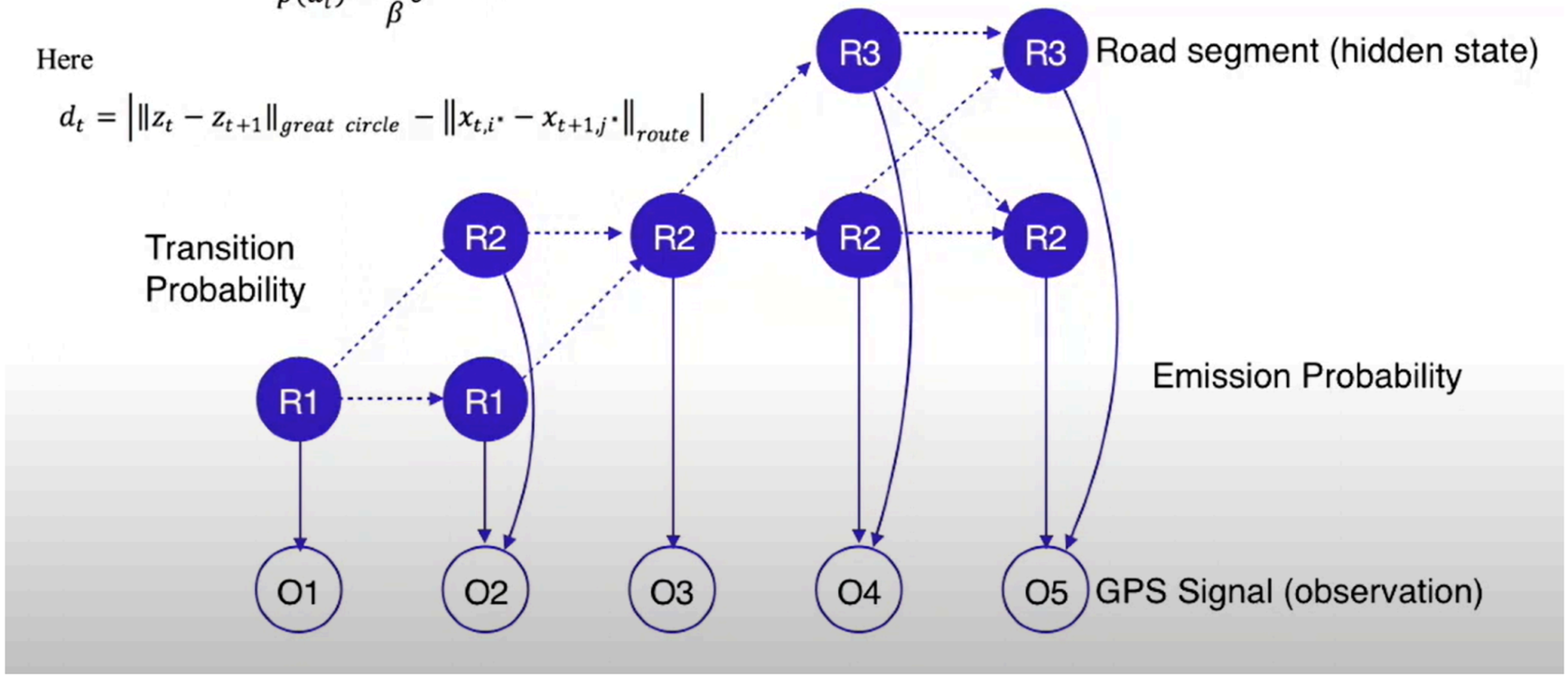
Emission Probability:

For each GPS point, Valhalla computes how likely it is to belong to nearby road segments.

$$p(d_t) = \frac{1}{\beta} e^{-d_t/\beta}$$

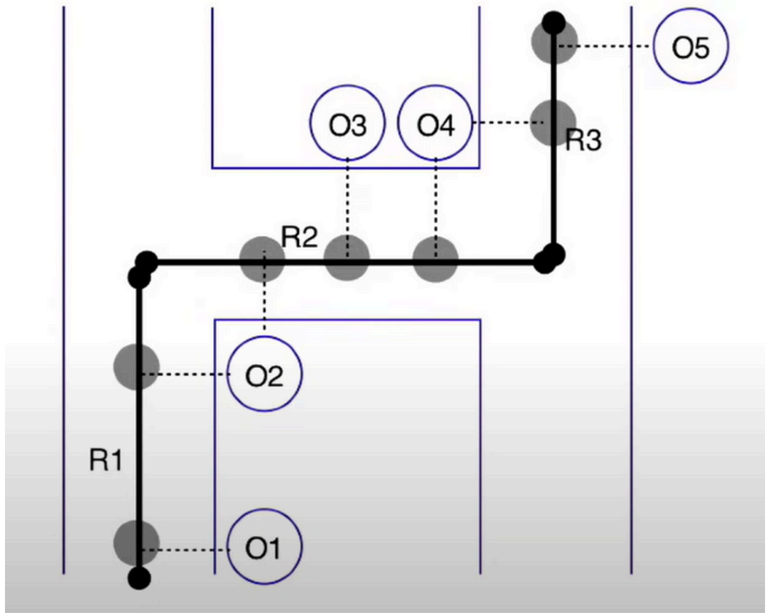
Here

$$d_t = \left| \|z_t - z_{t+1}\|_{great\ circle} - \|x_{t,i^*} - x_{t+1,j^*}\|_{route} \right|$$



Transition probability:

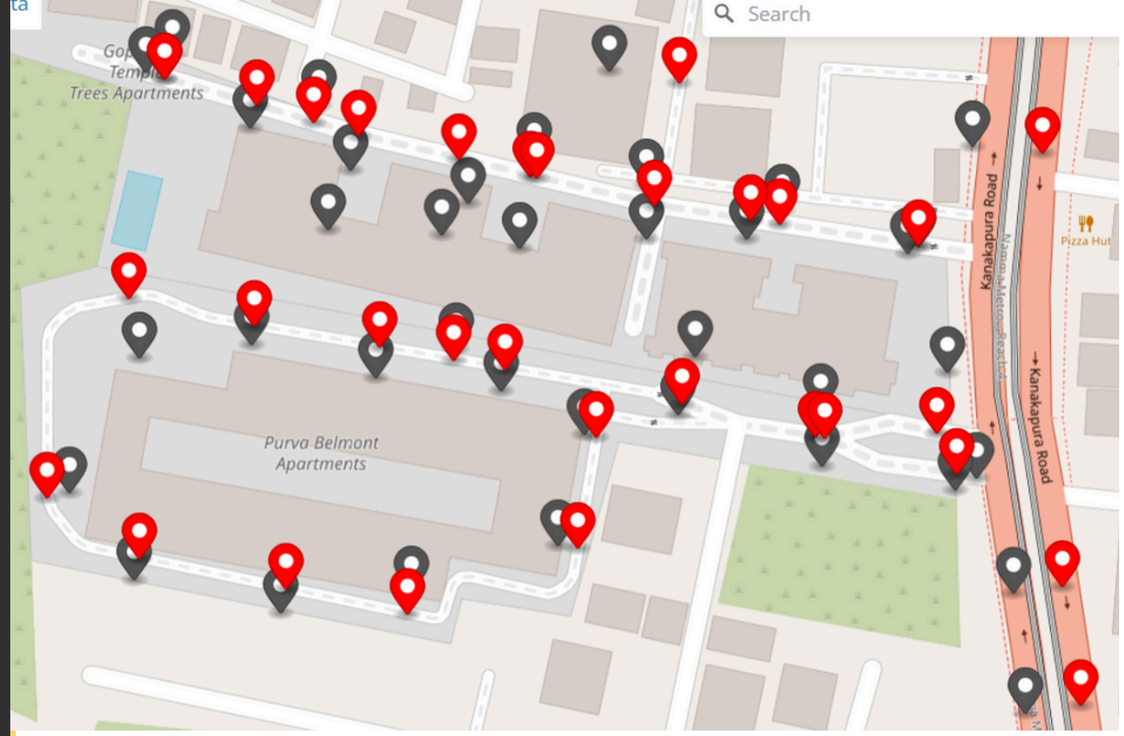
It measures how realistic the movement is between two GPS points based on the road network. It compares the aerial distance between GPS traces vs. route distance between road segments. Unrealistic road transitions get lower probability

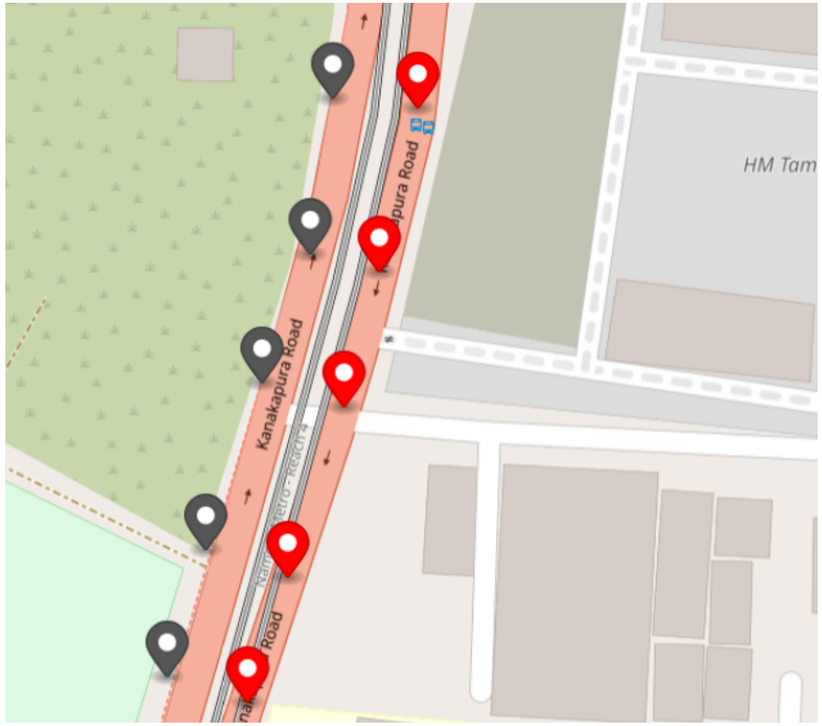


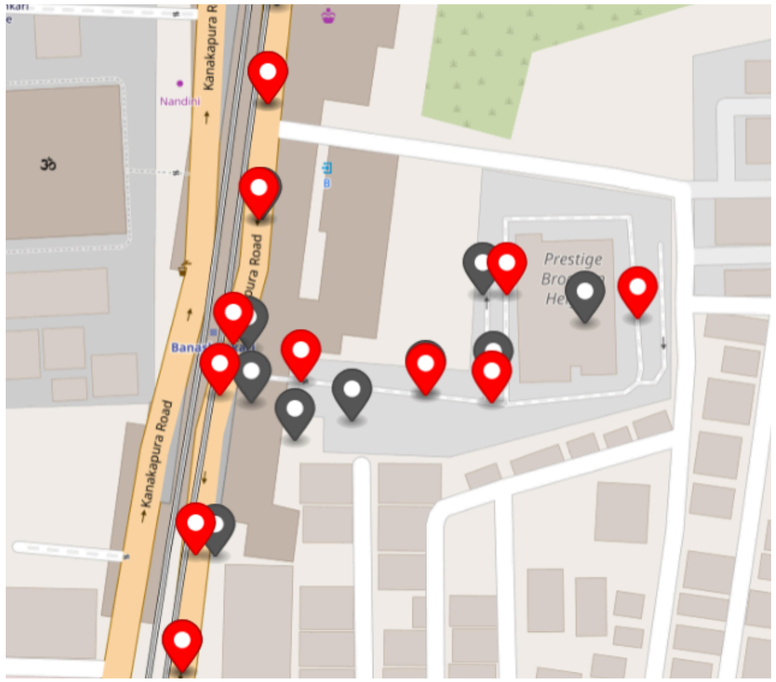
Viterbi is a dynamic programming algorithm used to find the most likely sequence of hidden states (roads) given a sequence of observed data (GPS points).

For each candidate:

1. Computes Emission Probability
2. Computes Transition Probability
3. Stores the max probability path leading to each candidate


Search
Trees Apartments
Purva Belmont
Apartments
Kanakapura Road
Pizza Hut


Kanakapura Road
HM Tam


Kanakapura Road
Nandini
Prestige
Kanakapura Road

logic for stoppage detection:

```
start = p1
end = p2

x = distance between points
y = time between points

If x > threshold_distance:
    calculated_time = x / average_speed

    If y > calculated_time:
        flag = 1
        start = end
        end = end + 1

Else:
    If y < threshold_time:
        end = end + 1
    Else:
        flag = 1
        start = end
        end = end + 1
```

FURTHER IMPROVEMENTS:

1. Traffic Data Integration
   - Traffic data (e.g., from OpenTraffic, Google Maps, TomTom) provides real-time information about road conditions such as traffic speeds, congestion, and incidents (e.g., accidents or road closures).
   - Valhalla can incorporate this traffic data to adjust travel times for different road segments.
2. Process Flow for Integration
   1. Collect Traffic Data: Obtain real-time or historical traffic data for specific roads using APIs (e.g., OpenTraffic API or Google Maps Traffic API).
   2. Adjust Valhalla's Routing: Valhalla uses the traffic data to modify the route planning. For example, if a road is congested, Valhalla will adjust the speed for that road segment.
   3. Route Calculation: When a user requests a route, Valhalla calculates the travel time based on traffic speeds and congestion. It then generates an optimized path.
   4. Stoppage Detection: Compare actual travel times (from GPS data) with expected travel times based on the adjusted traffic data. If the actual time exceeds the expected time, it's flagged as a stoppage (e.g., caused by traffic congestion).

FURTHER IMPROVEMENTS

3. Key Benefits

- Real-Time Traffic Adjustment: Routes are calculated based on real-time traffic conditions, ensuring optimal travel time and reduced congestion.
- Stoppage Detection: Detect delays by comparing real-time travel time with expected traffic-adjusted time to flag any stoppages or abnormal delays.
- Better Routing: With traffic data, Valhalla can dynamically reroute users around congested areas or incidents, improving user experience.

FURTHER IMPROVEMENTS

once we capture the speed of the vehicle :
For calculating the expected time, instead of using a fixed speed, the system will now calculate the expected travel time based on the distance between two consecutive points and the average speed between those points. This will ensure that the expected time more accurately reflects the actual movement along the route.