Documentation: GPX-to-JSON Normalizer

Auto-generated

July 31, 2025

1 Map Preprocessing and Graph Construction

To enable candidate edge detection and Hidden Markov Model (HMM)-based path inference, the road network must be represented as a graph. This section describes how OpenStreetMap (OSM) data is either dynamically fetched or loaded from disk and filtered for bicycle-compatible roads.

1.1 Input Format

The program accepts a JSON file consisting of sequential GPS observations. Each observation is a dictionary with spatial and contextual metadata such as latitude, longitude, elevation, timestamp, temperature, heart rate, cadence, speed, and heading.

```
{
  "lat": 38.821468,
  "long": -104.862572,
  "elv": 1886.6,
  "time": "2024-08-06T13:12:10+00:00",
  "atemp": "20",
  "hr": "104",
  "cad": "62",
  "speed": 0.0,
  "heading": 0.0
}
```

1.2 Graph Loading

The road network can be supplied in two ways:

1. **From a GraphML File:** If the user provides a file using `--graphml`, the graph is loaded via `osmnx.load_graphml()`.
2. **From OSM Bounding Box:** If no file is provided, the code calculates the bounding box of all valid GPS coordinates and queries OSM via `osmnx.graph_from_bbox()`.

1.3 Road Type Filtering

Only road types usable by cyclists are retained in the graph. This includes:

- `cycleway`
- `primary, secondary, tertiary`
- `residential, unclassified, service`

Edges not matching these categories are removed to improve routing accuracy and reduce computational overhead.

1.4 Caching

To avoid repeated API calls during development, OSM data is cached locally in a `./cache` folder.

1.5 Execution

Run the script with either:

```
# Load from GraphML file
python get_map_data.py test.json --graphml my_graph.graphml

# Download from OSM based on GPS bounding box
python get_map_data.py test.json
```