Self Localization via Path Shapes

Advanced Navigation of (Electric) Vehicles in Street Networks

Planning Energy-Efficient Routes

Taking into Account Loading Stations

Query Answering

1. Gathering Shape Data
   - Odometer → site
   - ESC → relative turning angles
   - Odometer sample into pieces of 1 m

2. MA - Massachusetts, GER - Germany
   - Shape-Preserving Dijkstra (SPD)
     - node length = path length

3. Shapes as Strings
   - IDEA: consider shape encoding as concatenated string, thes + angles
   - APPROACH: build generalized suffix tree (GST) on 'text'

4. Present-Day Localization Schemes

   - GPS
   - GSM
   - WPS

   - NEW: reference path shape stored

5. Now Scheme: Path Shapes
   - BASIC IDEA: use movement pattern to determine the own position in the street network
   - GIVEN: street network \( G(V,E) \), shape of driven trajectory
   - GOAL: find the (unique) starting vertex where the sequence of movements implied by the shape is possible ⇒ derive the own position in \( G \)

6. Advantages of Path Shapes
   - Fully autonomous
   - Consistent data quality
   - GPS-precise
   - Efficient query answering

Planning Energy-Efficient Routes

Motivation
   - Electric Vehicle (EV) is battery-powered
   - The cruising range is restricted by the battery’s capacity
   - Loading stations are spaces
   - Reloading takes several hours

Choosing the most energy-efficient route (instead of the shortest one)
   - is eco-friendly
   - allows to exploit the whole cruising range
   - might ease recharging institutes

Given
   - street network \( G(V,E) \), \( |V| = n, |E| = m \)
   - costs \( c : E \rightarrow \mathbb{R} \) representing energy consumption
   - M maximal battery load of the electric vehicle (EV)

Goal
   - Compute path p from \( s \in V \) to \( t \in V \) with minimal energy consumption for initial battery load \( f \leq M \)

Problems
   - 1. partly negative edge costs
   - 2. battery constraints

Solution
   - 1. Model battery constraints into FIFO edge cost functions
     - Johnson's Shifting Technique
   - 2. Johnson's Shifting technique to get rid of negative edge costs
     - Dijkstra applicable
   - 3. Apply speed-up technique
     - Constraint Hierarchies (CH)

Query Answering

1. Compute \( L_p \subseteq L \) set of reachable LSs from \( s \)
2. Compute \( L_p \subseteq L \) set of applicable LSs from \( L_p \)
3. Run BFS from \( L_p \) in \( G \) until the first node in \( L_p \) is visited
4. Backtrack respective path

Total run time \( O(n \log n + m) \)

Planning Energy-Efficient Routes

Motivation

Taking into Account Loading Stations

A Loading Station (LS) is a node \( t \in V \), that leads to a full battery \( b_t = M \), whenever it is visited

The set of all LSs is called \( L \subseteq V \)

Goal

Compute path from \( s \in V \) to \( t \in V \) with minimal number of necessary recharging events

Approach

Build auxiliary graph \( Q \) upon loading stations in a preprocessing step

\[ Q(L) = (L, L') \]
   - \( L \) set of loading stations
   - \( L' \subseteq E \) with \( (r,t) \in L' \) if \( r \) is reachable from \( t \)
   - \( r \in L \) if \( r \) is reachable from \( s \in V \), if there exists a path from \( v \) to \( s \) which always the battery constraints and requires no recharging intervals

Query Answering

1. Compute \( L_p \subseteq L \) set of reachable LSs from \( s \)
2. Compute \( L_p \subseteq L \) set of applicable LSs from \( L_p \)
3. Run BFS from \( L_p \) in \( G \) until the first node in \( L_p \) is visited
4. Backtrack respective path

Total run time \( O(n \log n + m) \)