AGH UNIVERSITY OF SCIENCE AND TECHNOLOGY

ICAISC Zakopane
International Conference on Artificial Intelligence and Soft Computing
June 1-5, 2014

An incremental map-matching algorithm based on Hidden Markov Model

Piotr Szwed and Kamil Pękala
AGH University of Science and Technology
Department of Applied Computer Science
e-mail: pszwed@agh.edu.pl

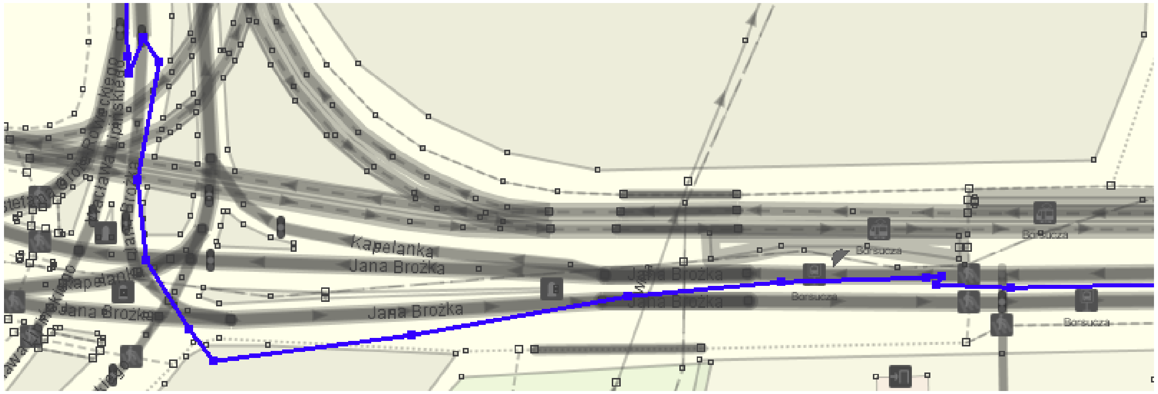
Agenda

1. Introduction
2. Related works
3. Processing steps
4. Hidden Markov Model set-up
5. Map matching algorithm
6. Experiments
7. Conclusions

Introduction

Map-matching aims at establishing the vehicle location on a road segment based on positioning data.

- Sensors (GPS, WiFi, cellular radios, odometers) return uncertain data.
- Current sensor reading and/or a number of past data can be used



Introduction 2

- Map-matching is used in Intelligent Transportation Systems (ITS)
  - Fleet management, vehicle tracking, navigation services
  - Traffic monitoring, congestion detection

Motivation

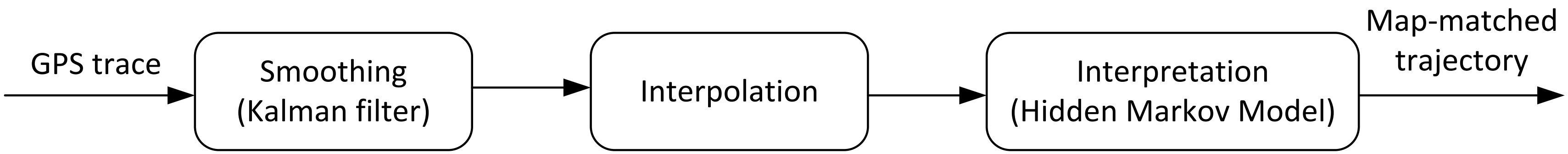
- Algorithm that can be used for
  - Tracking on-line individual vehicles
  - Monitoring traffic by crowd-sourcing
- The algorithm should be
  - *incremental* (calculate trajectory on arrival of new data)
  - as opposed to *global* (the whole trace is map-matched)

Related works

Map matching algorithms

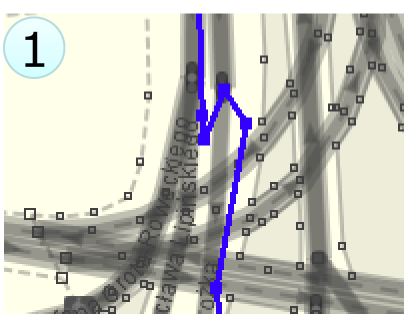
- *Geometrical* (point to curve or segment to curve matching) [White, Bernstein et al. 2000; Greenfeld 2002]
- *Topological*: utilize information about connections between road segments [Quddus, Ochieng et al. 2003].
- *Probabilistic*: use information on circular or elliptic confidence region associated with position reading [Ochieng, Quddus 2009]
- *Advanced*: Kalman filter, fuzzy rules, Particle filters (both topological and probabilistic) [Fu, Li et al. 2004; Gustafsson, Gunnarson et al., 2002]
- *Incremental* algorithms using tree-like structure [Marchal, Hackney et al. 2004; Wu, Zhu et al. 2007]
- Global algorithms based on *Hidden Markov Models* [Newson, Krumm 2009; Thiagarajan, Ravindranath et al. 2009]

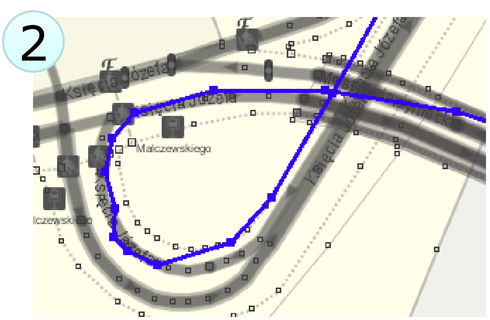
Processing steps

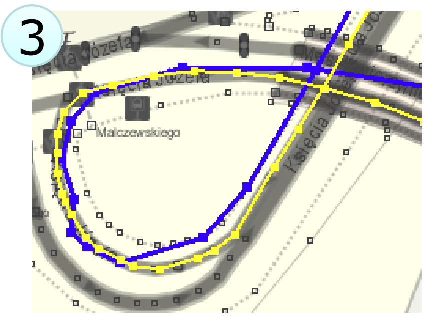


The system is organized into a **pipeline**:

1. Smoothing with Kalman filter (removal of outliers)
2. Interpolation (matching the map scale)
3. Interpretation (actually map-matching)







Hidden Markov Model

Hidden Markov Model: $\lambda = (Q, A, O, P_t, P_o, q_0)$

Q – set of states

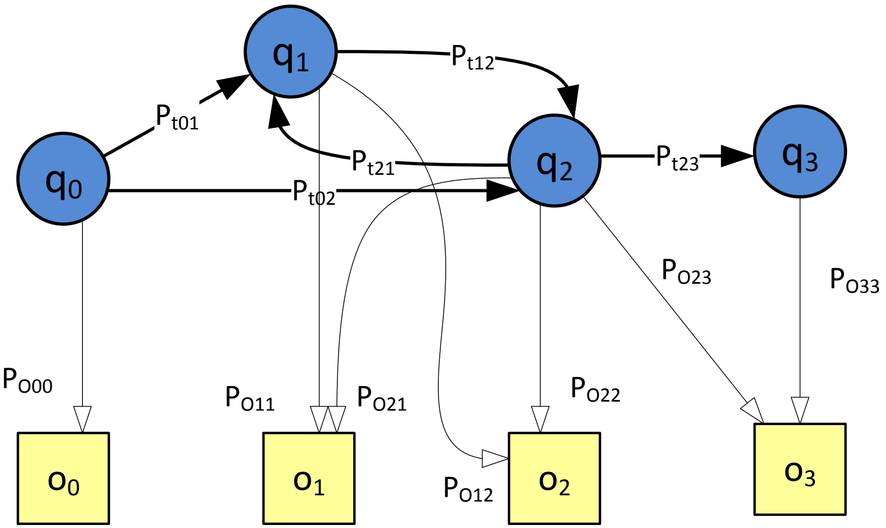
$A \subset Q \times Q$ – set of arcs

O - set of observations

$P_t : A \rightarrow (0,1]$ – state transition probability

$P_o : Q \times O \rightarrow [0,1]$ – emission probability

q_0 - initial state

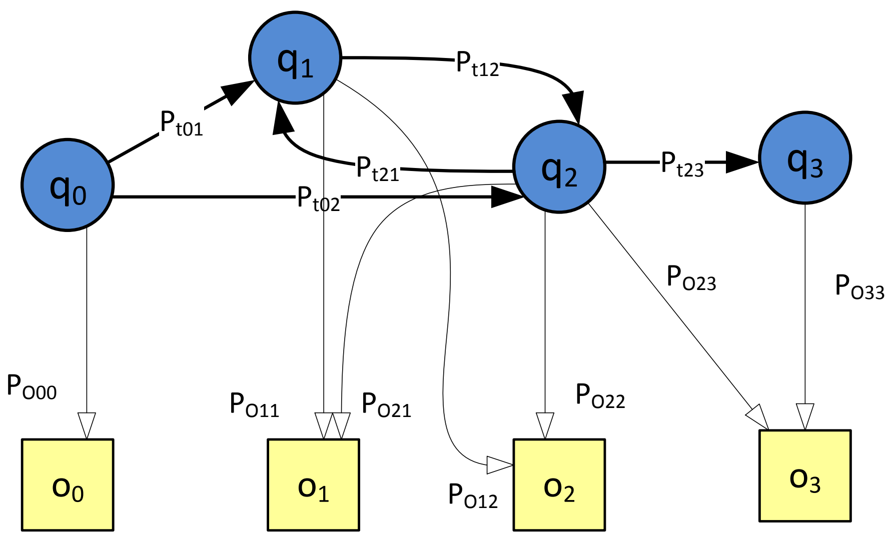


Hidden Markov Model – Decoding

Decoding problem:

- given a sequence of observations $o_{i1}, o_{i2}, \dots, o_{in}$
- find the most probable sequence of hidden states $q_{i1}, q_{i2}, \dots, q_{in}$

Resolved with Viterbi algorithm



Idea of application to map-matching:

- observations: readings from a location sensor (GPS, WiFi)
- hidden states: road segments

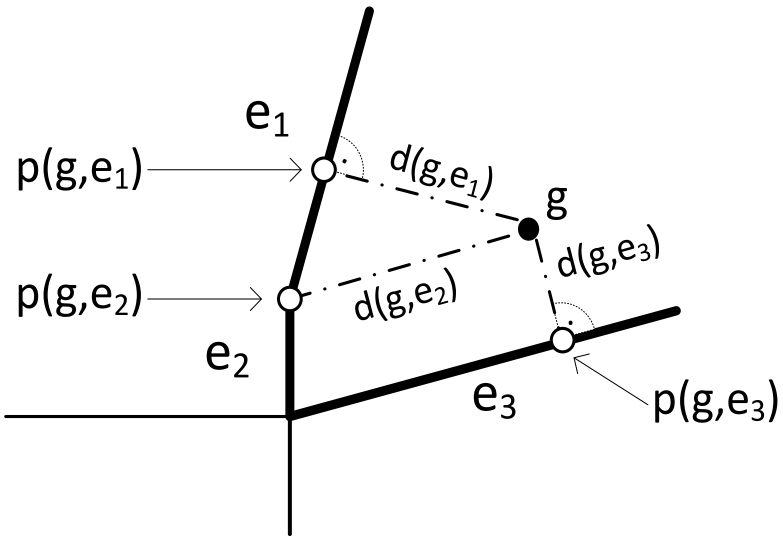
HMM model setup 1

Road network model

$G = (V, E, I)$, where

- $V \in \boldsymbol{R} \times \boldsymbol{R}$ – node (longitude, latitude)
- $E \subset V \times V$ – edge (straight segment)
- $I \subset E \times E$ – specify forbidden maneuvres at junctions

Projection of a point g on a segment $\boldsymbol{e} = (\boldsymbol{v_b}, \boldsymbol{v_e})$

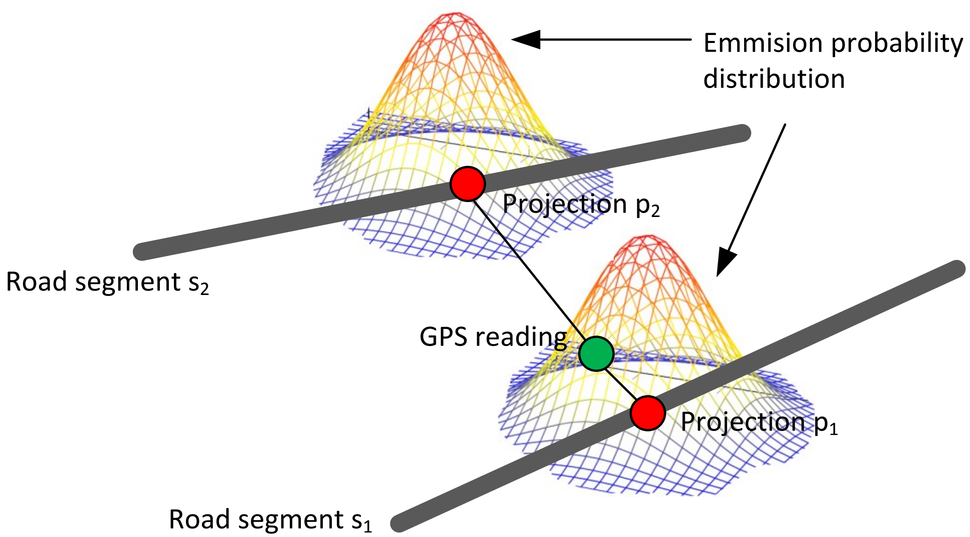


$$p(e, g) = \underset{g' = v_b + t(v_e - v_b) \wedge t \in [0,1]}{\arg\min} d(g, g')$$

$d(g, g')$ – distance (haversine formula)

HMM model setup 2

- HMM state $q = (e, p, i)$ e- road segment, p – projection point, i – sequence number
- Transition probability $P(q_1, q_2)$ – equal at junctions, low probability for forbidden meneuvres, dead recokonning on speed



- Observation $o = (lon, lat, time)$
- Normal distribution for the emmision probability:

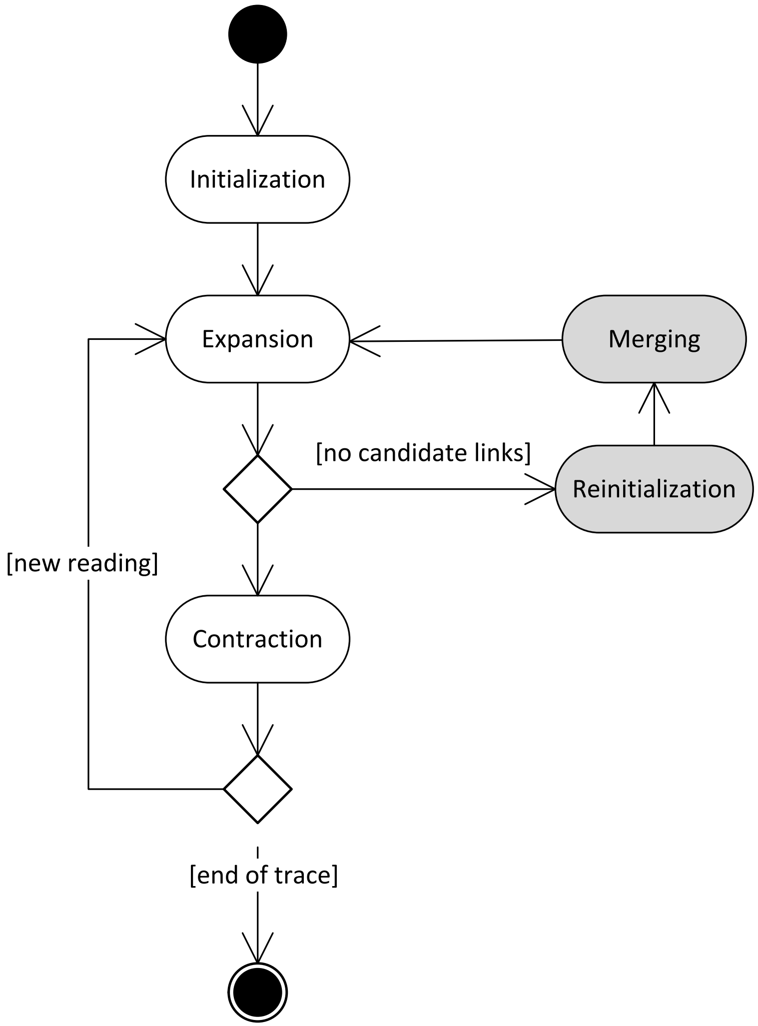
$$P(x, y) = \frac{1}{D} e^{-k\left(\left(x - x_p\right)^2 + \left(y - y_p\right)^2\right)}$$

(x_p, y_p) - projection point

k – depends on a sensor

$D = \iint_{-\infty}^{\infty} P(x, y) dx\, dy$ – normalization

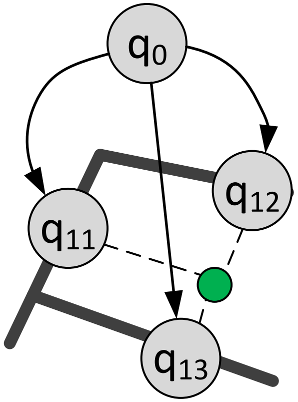
Map matching algorithm

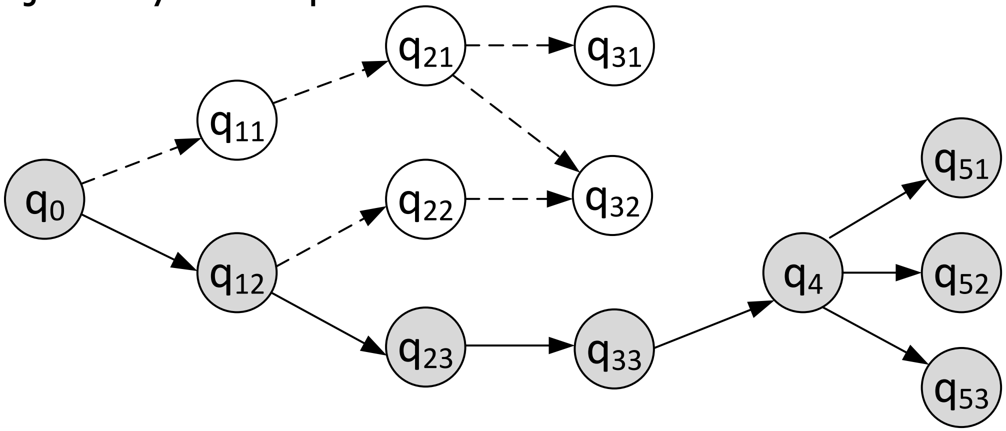


- Input: seqence of observations $\omega = (o_i\colon i = 1, n)$
- Internal data: a seqence of HMMs $\Lambda = (\lambda_i : i = 1, n)$
- Output: sequence of HMM states (projection point on road segments)

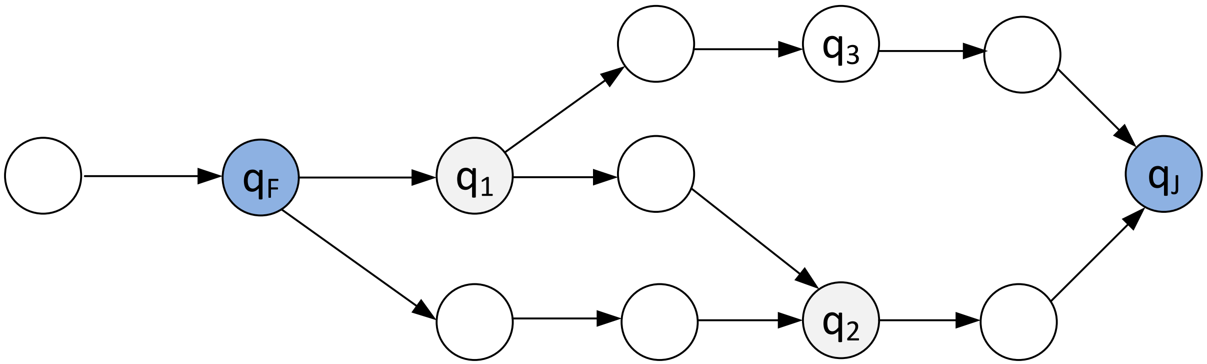
Map matching algorithm

- **Initialization**: the first model λ_1 is built by linking an artificial state with projections of o_1 on road segments.
- **Epansion**: observation o_i is projected on road segments. Then, obtained new states are linked with the last states (segments) from λ_{i-1}
- **Contraction**:
  - orphan nodes without successors are removed
  - the HMM root is moved forward and a next part of the trajectory is output.





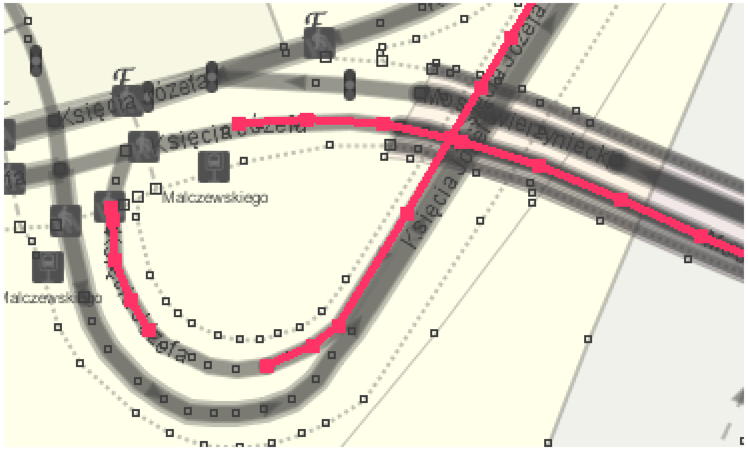
Handling joins



- A state q_J is a join, if it has at least two different predecessors.
- Presence of a join indicates that the vehicle positions were assigned to parallel roads that finally joined.
- If HMM contains a state (here q_F), from which all traces lead to q_J, the Viterbi algorithm is applied to the subgraph.
- States lying beyond the computed path are removed.

Handling special situations

- If no candidate segments are found in i-th iteration, expansion fails (reason: noise and/or map density).
- The algorithm performs reinitialization and creates new model λ_{i0}
- Depending on application, models λ_{i-1} and λ_{i0} can be:
  - merged (vehicle tracking)
  - left unconnected (traffic monitoring)



Two reinitializations for a corrupted trace: half-sampled and noised (20m)

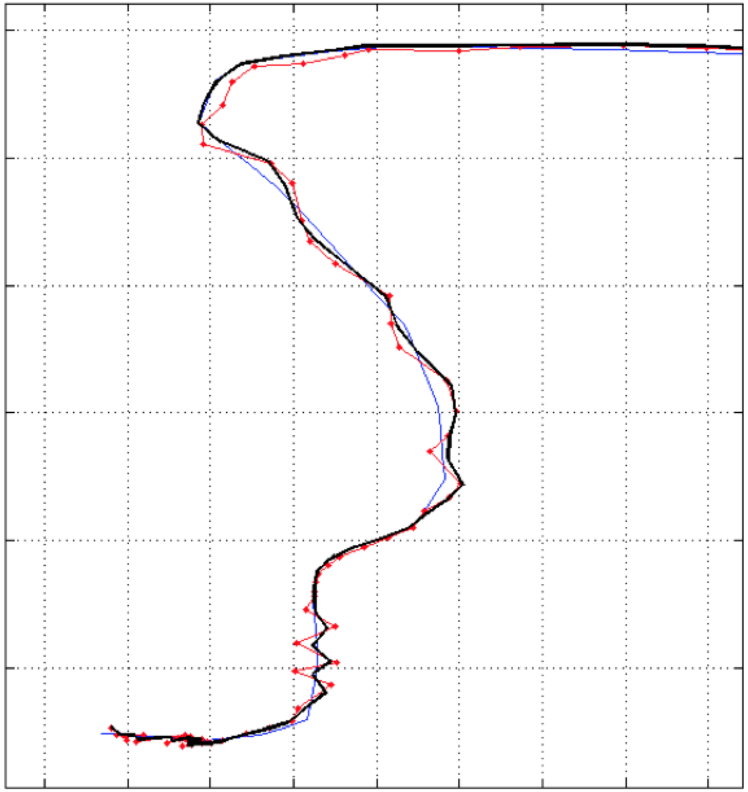
Experiments 1

Dataset

- Map of Kraków, data source: Open Street Map (OSM)
- 20 GPS traces registered with EasyTrials GPS software on iPhone 5 (148.6km, 4482 readings)

Kalman filter

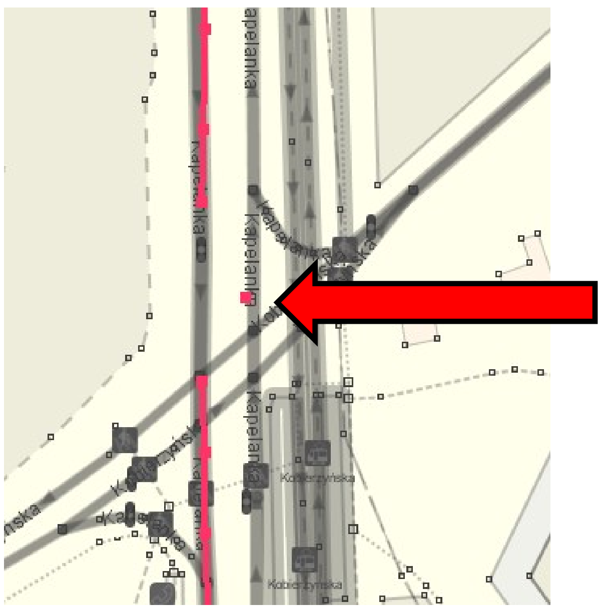
- Two distinct second order filters processing separately noise for longitude and latitude components
- State variables: position and velocity
- Initial parameters determined empirically with Matlab.

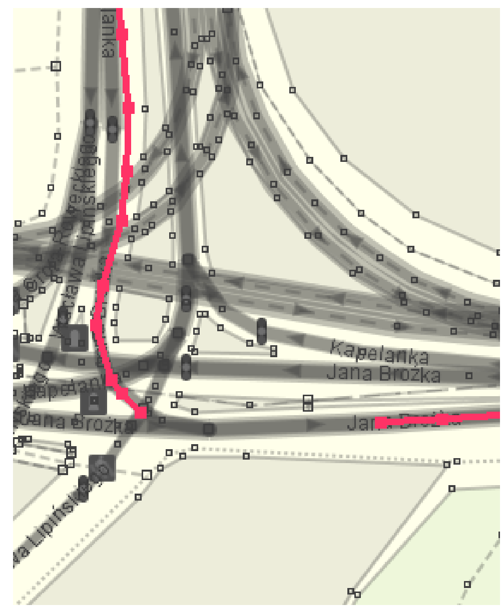


Experiments 2 (criterion)

Main criterion: **number of reinitializations**

- Unrecoverable errors occured in about 30% of the cases, in which the algorithm was forced to perform reinitialization.
- Reitialization usually takes more time than the normal algorithm processing step





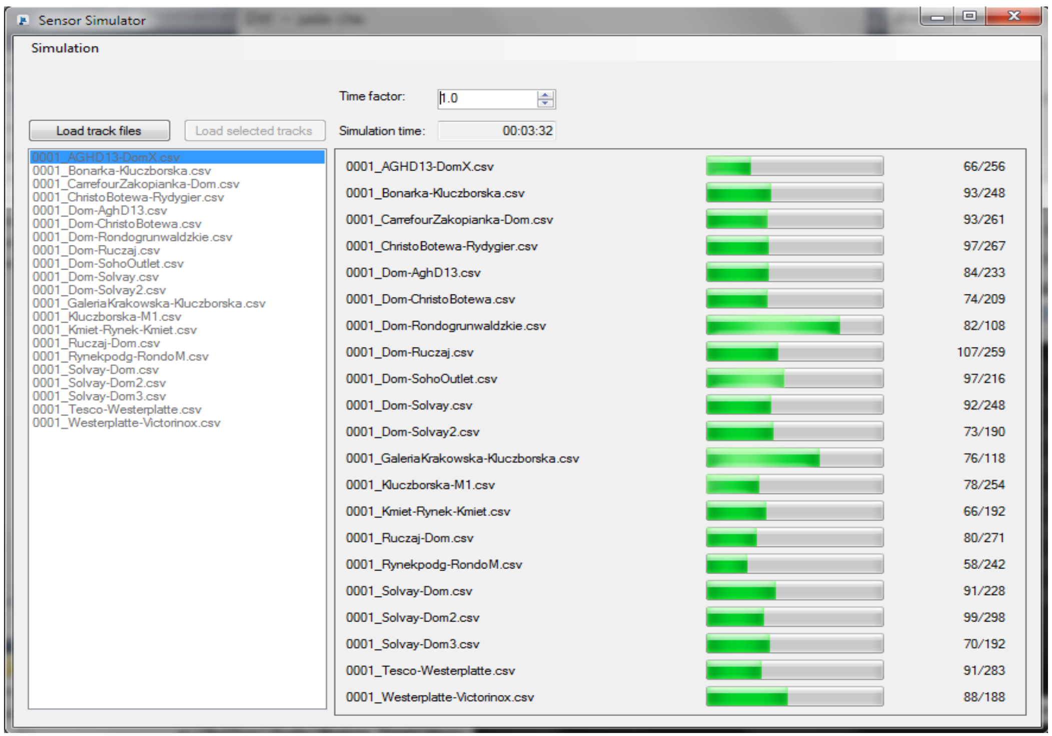
Interpretation of half-sampled path with artificial noise (20m): unrecoverable (left) and recoverable (right)

Experiments 3 (accuracy)

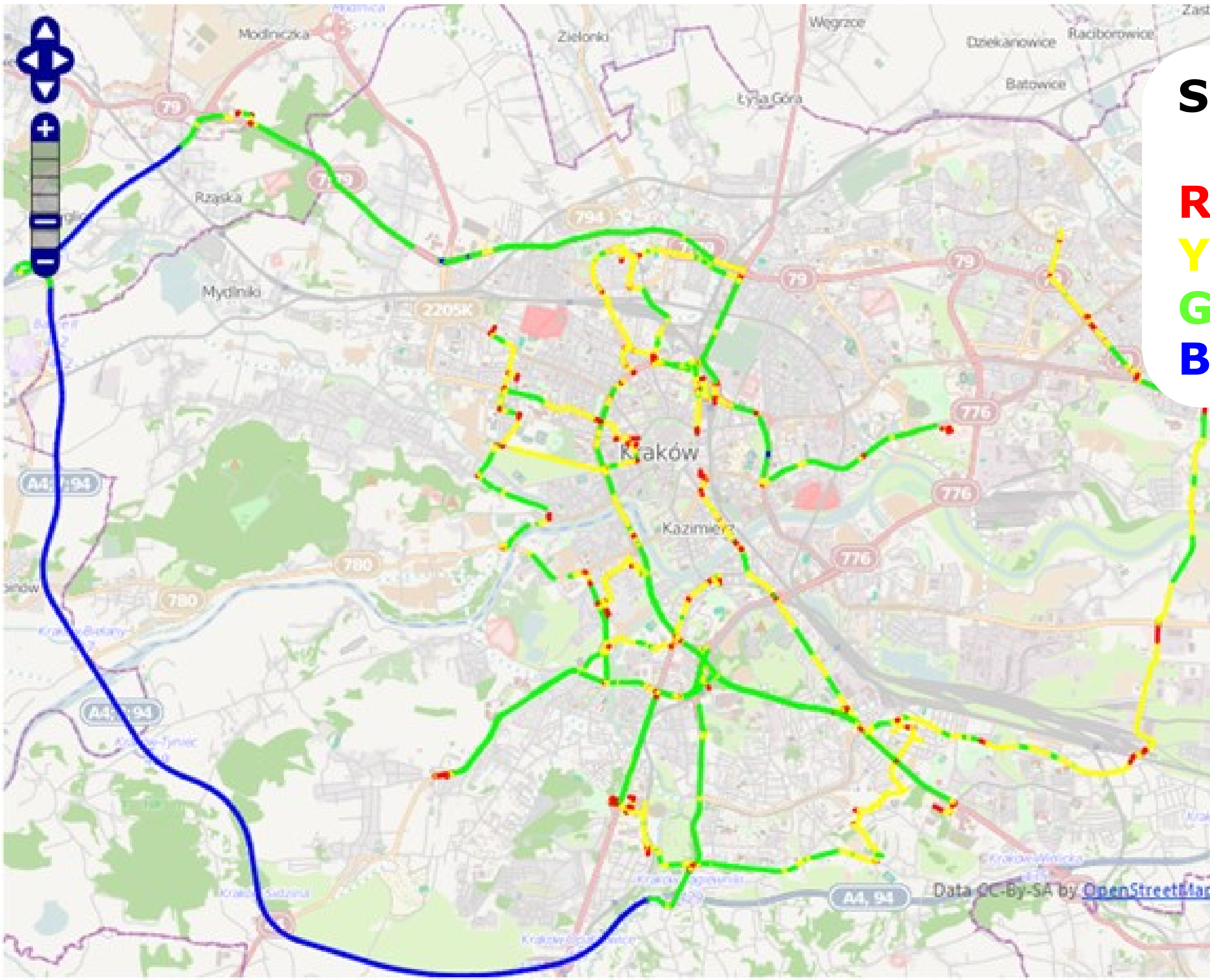
	RI - Number of reinitializations	RI/sample	Av. distance between RI
Normal	24	0.005	6.18 km
Noise (20m)	73	0.016	2.03 km
HS (Half-sampled)	23	0.005	6.44 km
HS+Noise	45	0.010	3.29 km

Experiments 4 (performance)

- Prototype system implemented in C#
- Capable of processing 20 simultaneous feeds with speed-up by 50



Speed map



Speed km/h

Red [0,20)
Yellow [20,50)
Green [50,90)
Blue: [90,∞)

Conclusions

Map-matching algorithm

- Based on Hidden Markov Model
- In each iteration HMM is
  - expanded by adding new states (projections on road segments)
  - contracted to output a next part of a vehicle trajectory.
- Structure of HMM forms in most cases a tree [Wu, Zhu et al. 2007] but parallel roads are supported
- Viterbi algorithm used only, if parallel roads are encountered
- The algorithm is *incremental* (required for real-time services)

Thank you