Fast HMM Map Matching for Sparse and Noisy Trajectories

Hannes Koller, Peter Widhalm,
Melitta Dragaschnig, Anita Graser

PETER WIDHALM
Mobility Department
Dynamic Transportation Systems

T +43(0) 50550-6655 | F +43(0) 50550-6439
peter.widhalm@ait.ac.at | http://www.ait.ac.at

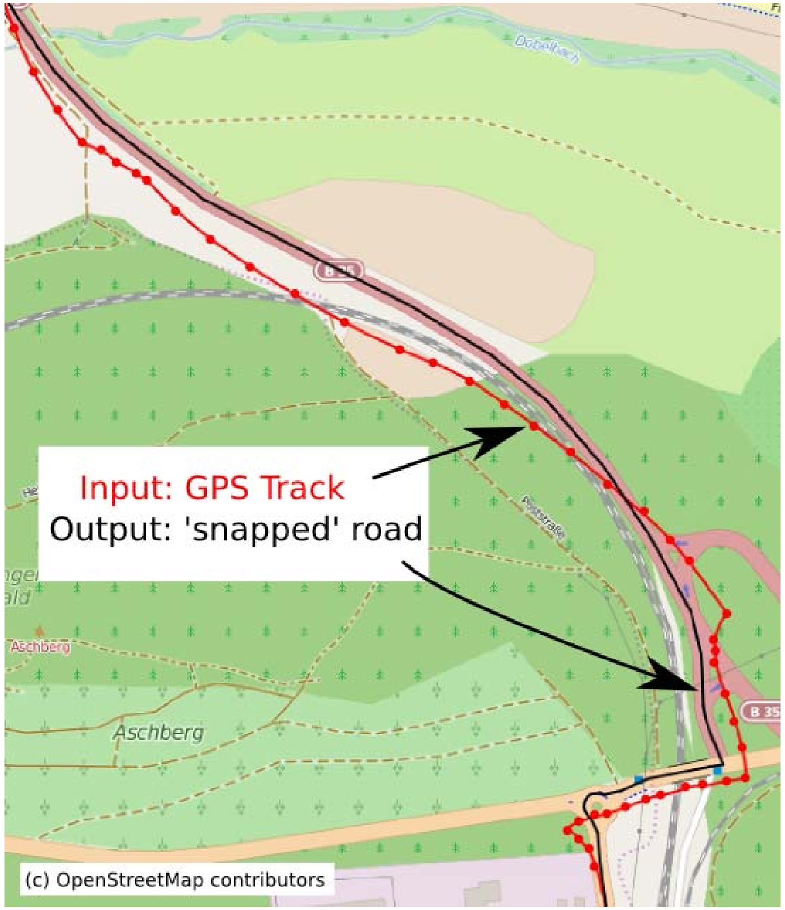
HANNES KOLLER
Mobility Department
Dynamic Transportation Systems

T +43(0) 50550-6674 | F +43(0) 50550-6439
hannes.koller@ait.ac.at | http://www.ait.ac.at

What is „map matching“?

- Wikipedia[1]:

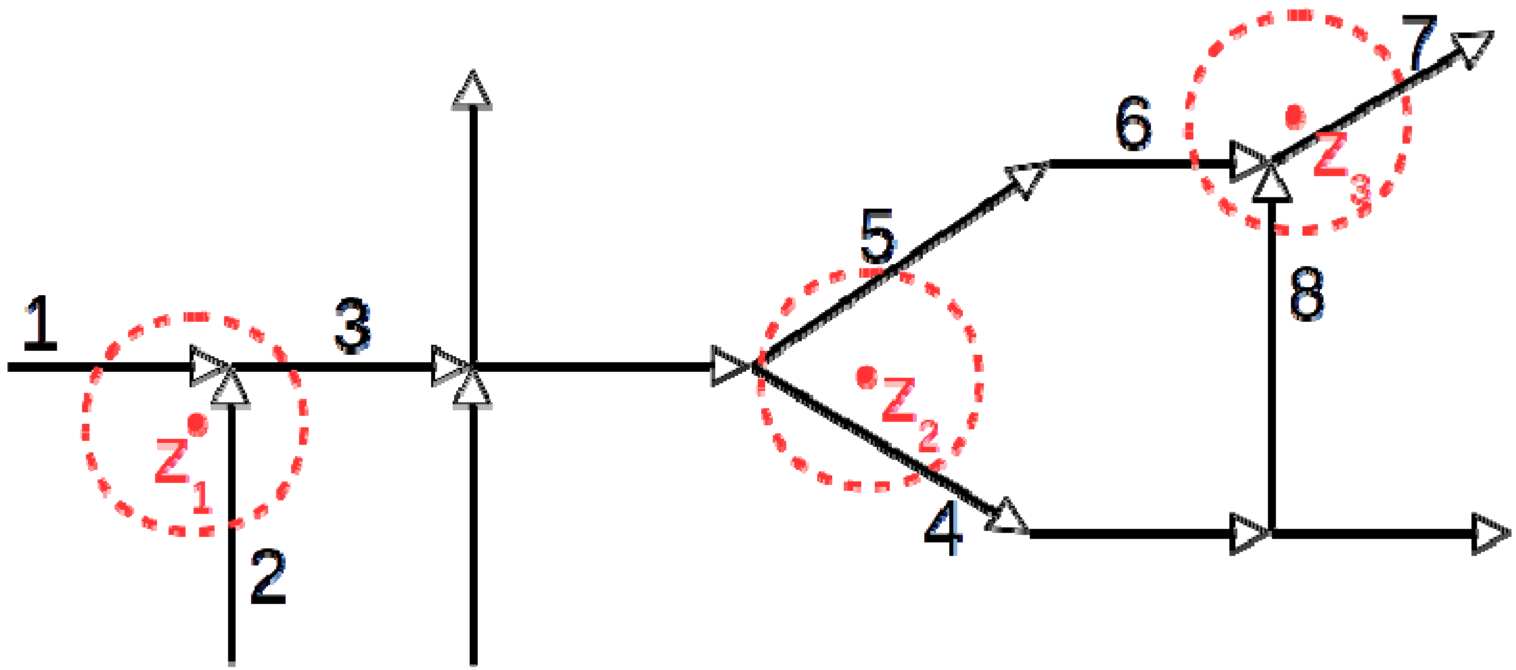
> “**Map matching** is a technique in GIS that associates a sorted list of user or vehicle positions to the road network on a digital map.
>
> The main purposes are to track vehicles, analyze traffic flow and finding the start point of the driving directions.”



[1] https://en.wikipedia.org/wiki/Map_matching

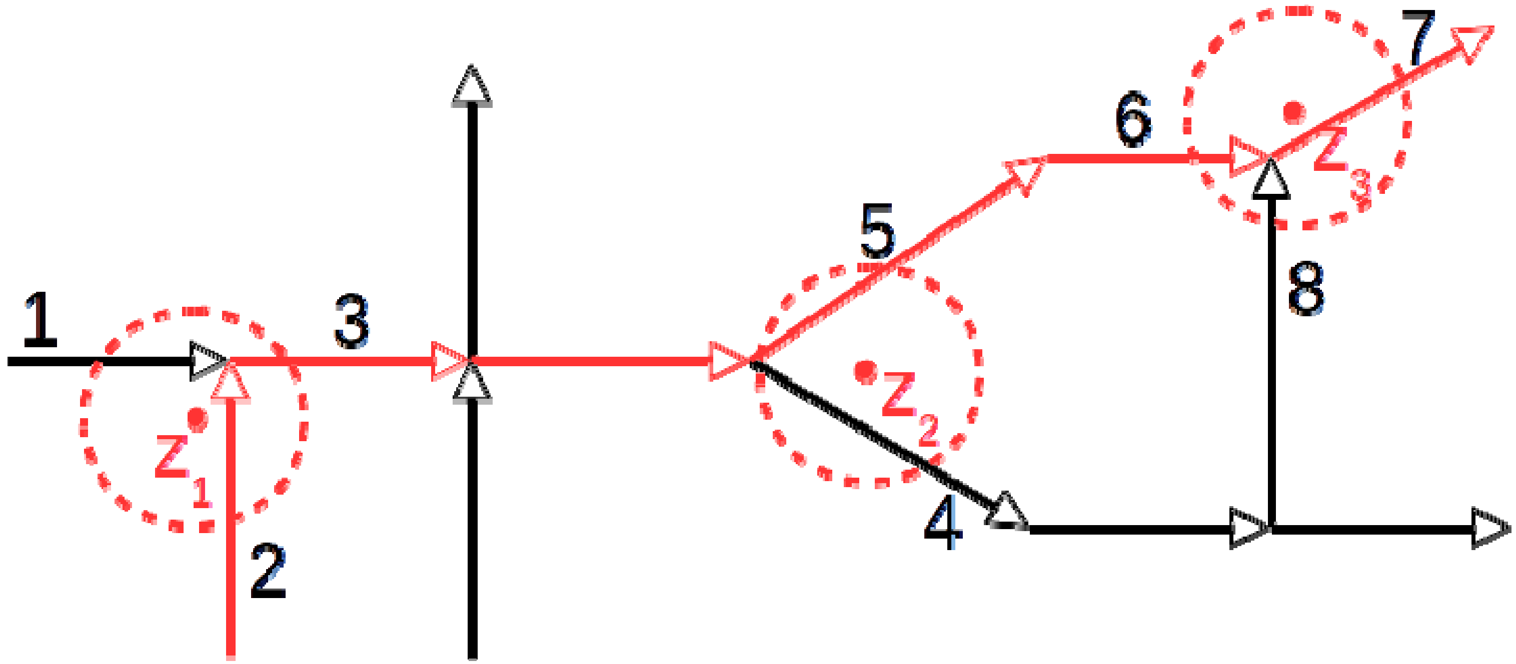
Why can map matching be difficult?

- Noisy measurements, a dense road network and sparse sampling in time make this task difficult
- Simple „point-to-curve“ matching quickly becomes insufficient



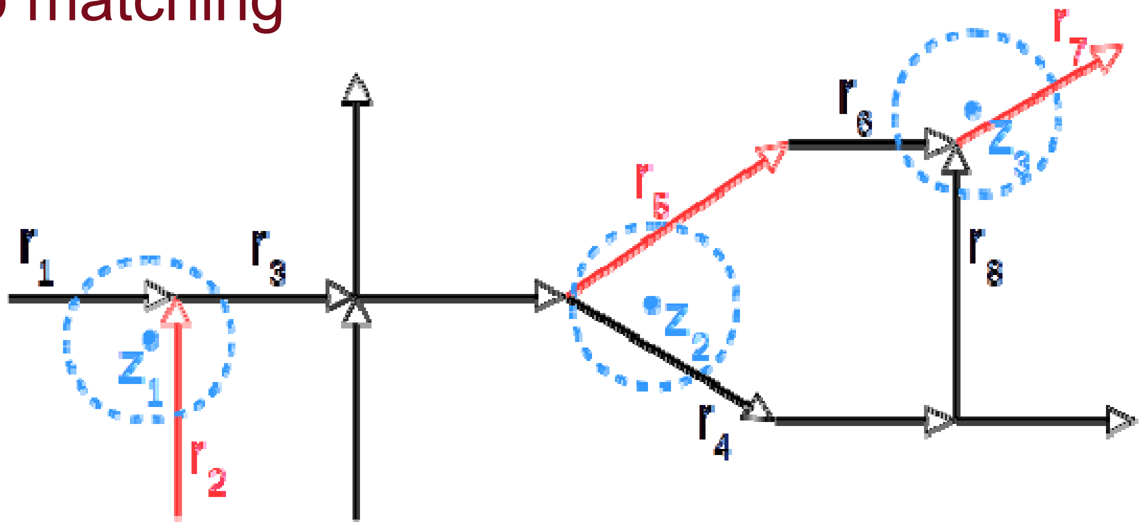
State-of-the-art solutions

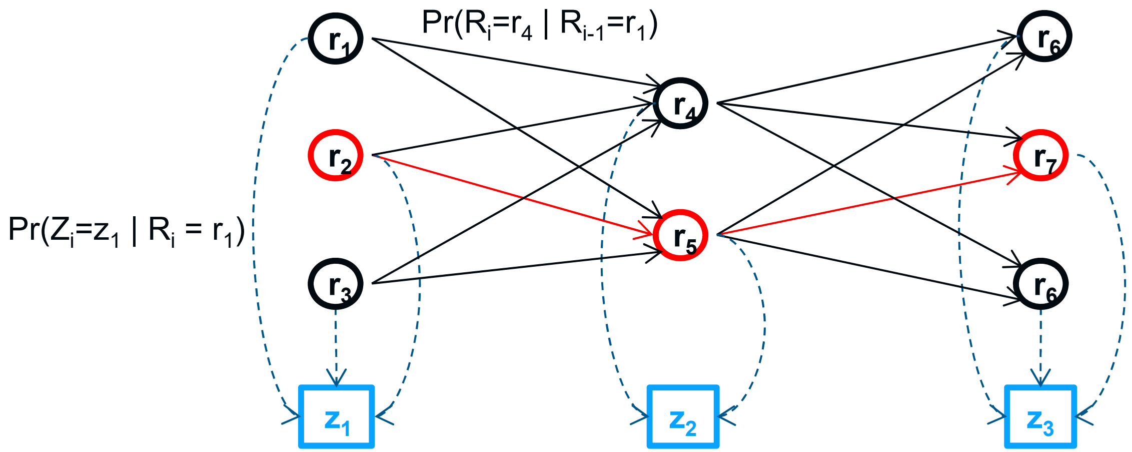
- use *information from* ***<u>all trajectory points</u>*** and *choose the* ***<u>most likely path</u>*** through the road network given the available position estimates
- e.g. Hidden Markov Model map matching [Newson and Krumm, 2009][2]



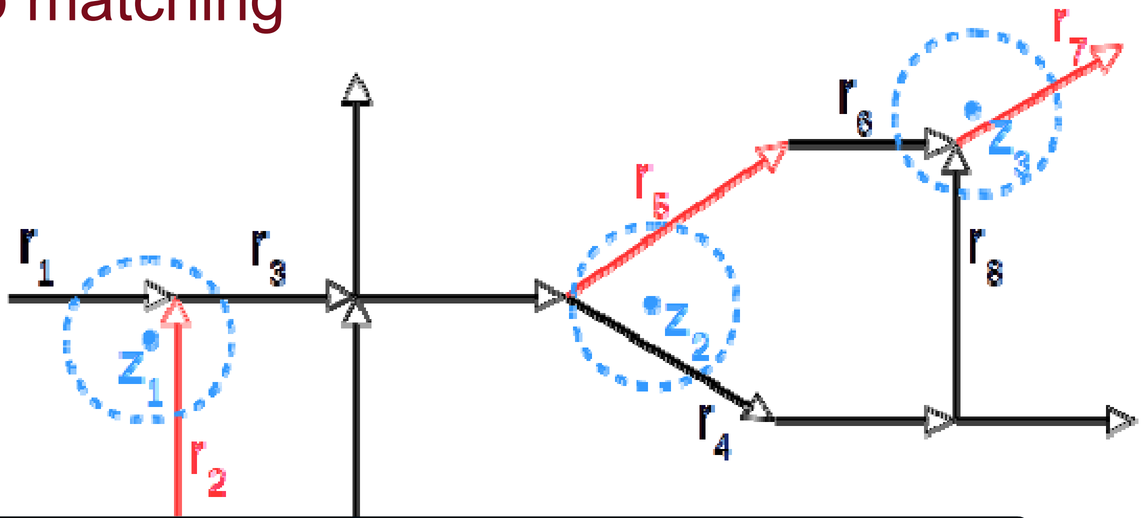
[2] Paul Newson and John Krumm. Hidden markov map matching through noise and sparseness. In Proceedings of the 17th ACM SIGSPATIAL International Conference on advances in Geographic Information Systems, pages 336–343, 2009.

HMM map matching

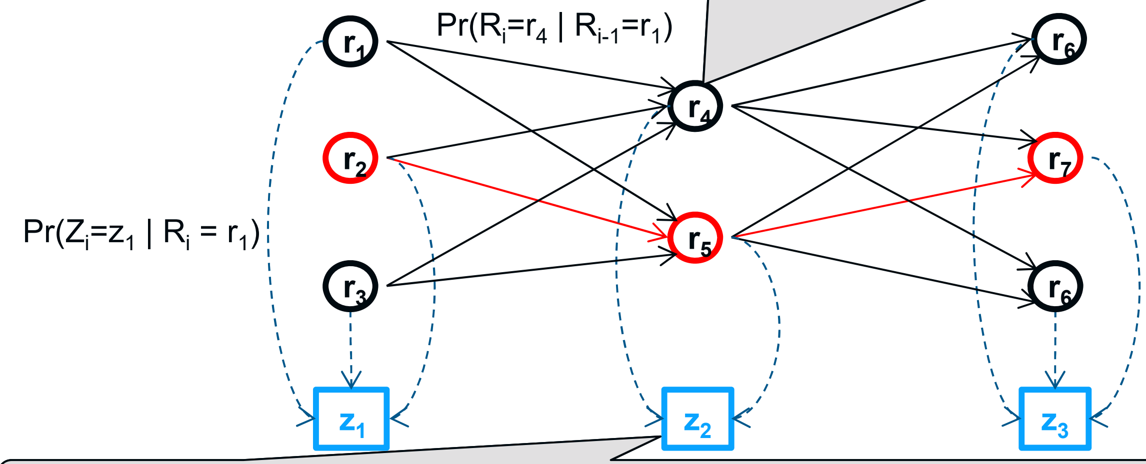




HMM map matching

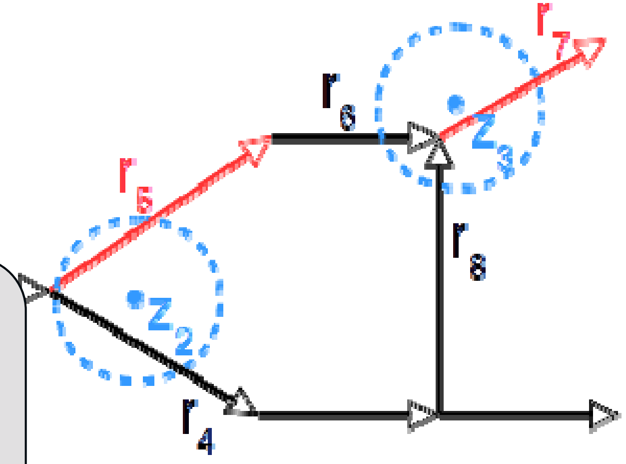


hidden states ***R*** = (R_1, …, R_n) ➔ **roads in the network**



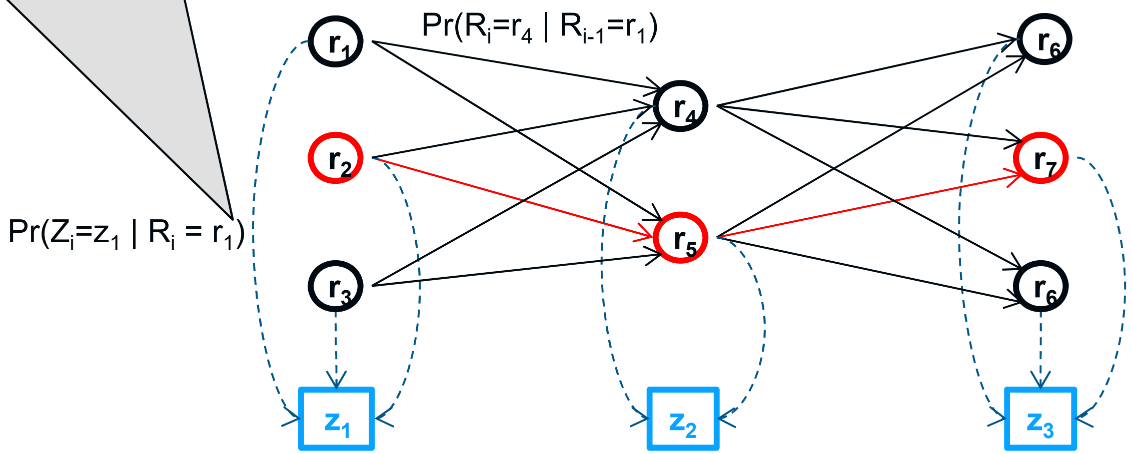
observable variable ***Z*** = (Z_1, …, Z_n) ➔ **position measurements** (e.g. GPS)

HMM map matching

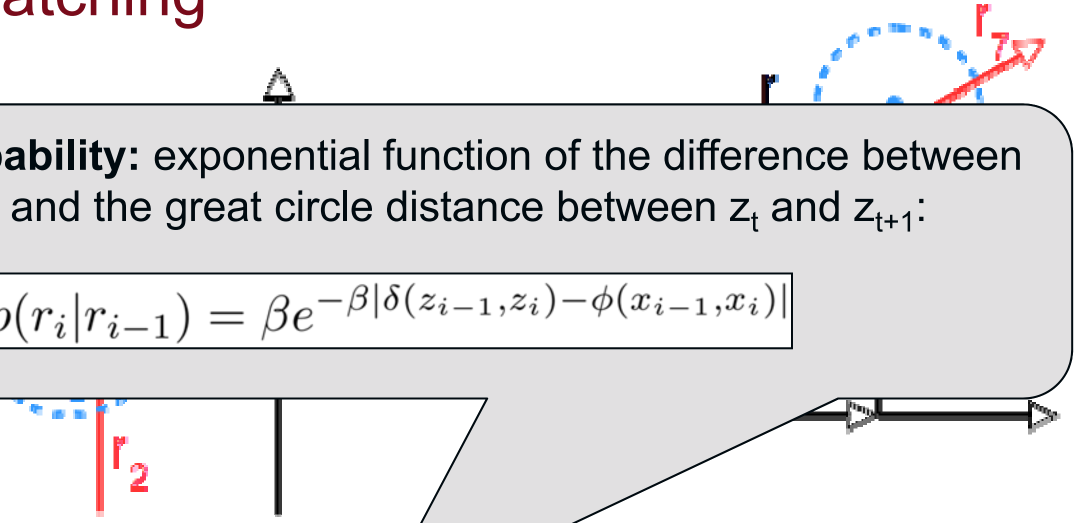


zero-mean Gaussian positioning error →

$$p(z_i|r_i) = \frac{1}{\sqrt{2\pi}\sigma_z} e^{-0.5\left(\frac{\delta(z_i,x_i)}{\sigma_z}\right)^2}$$

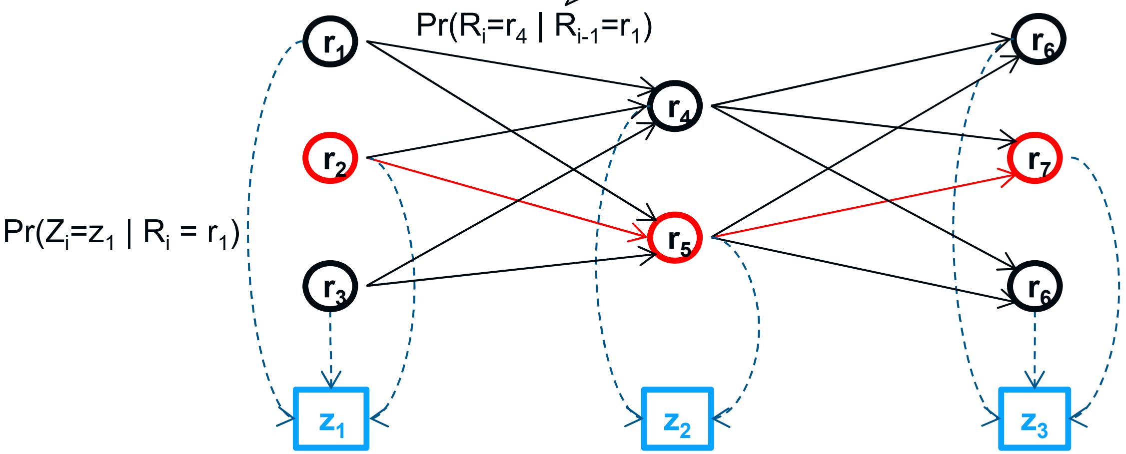


HMM map matching

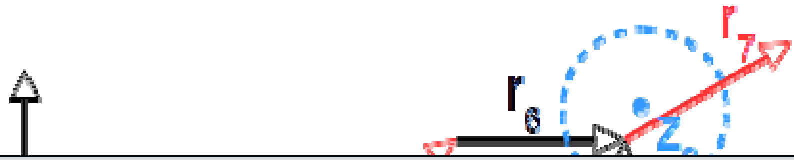


transition probability: exponential function of the difference between the route length and the great circle distance between z_t and z_{t+1}:

$$p(r_i|r_{i-1}) = \beta e^{-\beta|\delta(z_{i-1},z_i)-\phi(x_{i-1},x_i)|}$$



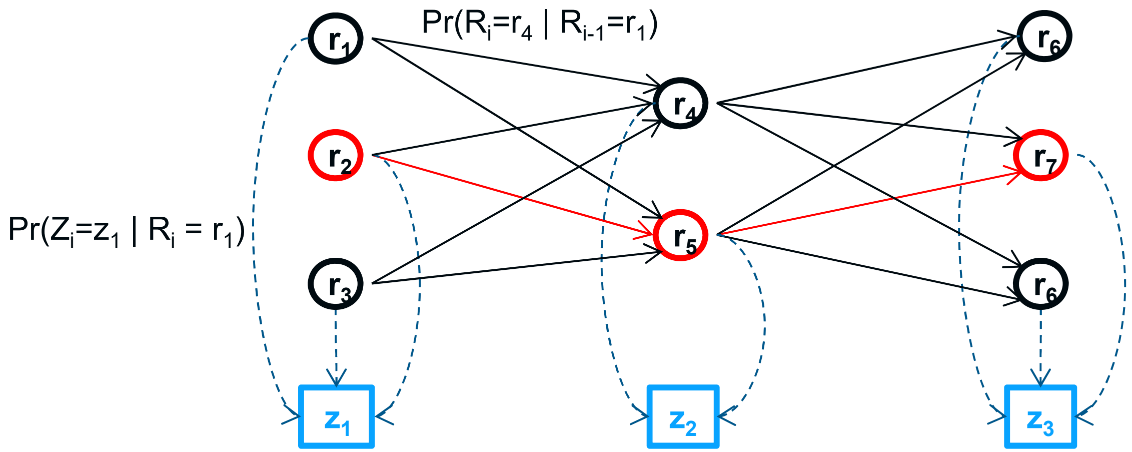
HMM map matching



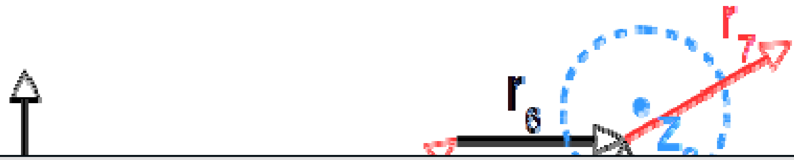
joint distribution of position measurements ***Z*** and roads in the network ***R***:

$$\boxed{p(R,Z) = p(z_0|r_0)p(r_0) \prod_{i=1..n} p(z_i|r_i)p(r_i|r_{i-1})}$$

➔ **maximize using *Viterbi algorithm!***



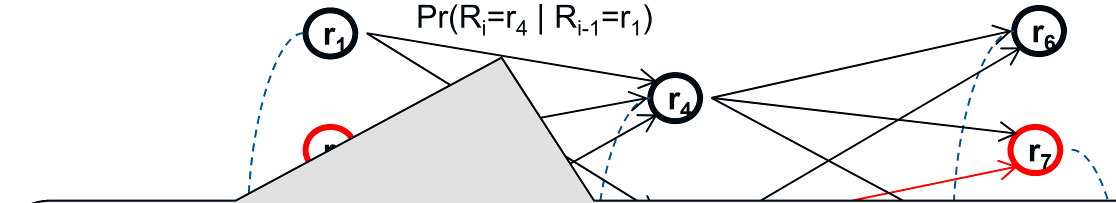
HMM map matching



joint distribution of position measurements ***Z*** and roads in the network ***R***:

$$p(R, Z) = p(z_0|r_0)p(r_0) \prod_{i=1..n} p(z_i|r_i)p(r_i|r_{i-1})$$

➔ **maximize using *Viterbi algorithm!***



The transition probability calculation requires a shortest path routing between each pair of candidate roads, which is a ***computationally expensive operation!***

→ **performance bottleneck!**

Improving run-time

- Previous approaches:
  - **parallelize** the computation of measurement and transition probabilities using **multi-threading** [Song et al., 2012][(3)]
  - determine paths from a candidate road to all of its successors **with a single execution of Dijkstra's algorithm** to reduce the number of required shortest-path routings from *nm* to *n* [Wei et al., 2012][(4)]
- **Our approach:**
  - reduce number of shortest-path routings by **replacing Viterbi algorithm** with bidirectional Dijkstra algorithm
  - complementary to previous approaches!

(3) R.Song, W. Lu, W.Sun. Quick Map Matching Using Multi-Core CPUs. ACM SIGSPATIAL GIS, 2012

(4) Hong Wei, Yin Wang, George Foreman Fast viterbi mapmatching with tunable weight functions ACM SIGSPATIAL GIS, 2012

Replacing Viterbi algorithm

- **Viterbi**
  - standardard algorithm to compute most likely sequence of states ***R*** given observations ***Z***
  - requires a full matrix of transition probabilities! → expensive!
- **bidirectional Dijkstra's algorithm**
  - well-known algorithm for minimum cost (e.g. shortest-path) routing
  - evaluates the costs of a node and its outgoing edges only when it arrives at this node during search!
  - in most cases only a fraction of all nodes needs to be visited before the minimum cost path is found [Nicholson, 1966](5)

(5) T.A.J. Nicholson Finding the shortest route between two points in a network The Computer Journal, Vol. 9, Nr. 3,S. 275-280, 1966.

Replacing Viterbi algorithm

- **Viterbi**

Maximize:

$$p(R, Z) = p(z_0|r_0)p(r_0) \prod_{i=1..n} p(z_i|r_i)p(r_i|r_{i-1})$$

- **bidirectional Dijkstra‘s algorithm**

Minimize:

$$C(R, Z) = \sum_{i=0..n} c_{node}(r_i, z_i) + c_{edge}(r_i, r_{i-1})$$

Replacing Viterbi algorithm

- **Viterbi**

Maximize:

$$p(R,Z) = p(z_0|r_0)p(r_0) \prod_{i=1..n} p(z_i|r_i)p(r_i|r_{i-1})$$

- **bidirectional Dijkstra's algorithm**

Minimize:

$$C(R,Z) = \sum_{i=0..n} c_{node}(r_i, z_i) + c_{edge}(r_i, r_{i-1})$$

$$-\log p(R,Z) = -\log p(z_0|r_0) - \log p(r_0) + \sum_{i=1..n} -\log p(z_i|r_i) - \log p(r_i|r_{i-1})$$

Replacing Viterbi algorithm

- **Viterbi**

Maximize:

$$p(R,Z) = p(z_0|r_0)p(r_0) \prod_{i=1..n} p(z_i|r_i)p(r_i|r_{i-1})$$

- **bidirectional Dijkstra's algorithm**

Minimize:

$$C(R,Z) = \sum_{i=0..n} c_{node}(r_i,z_i) + c_{edge}(r_i,r_{i-1})$$

$$-\log p(R,Z) = -\log p(z_0|r_0) - \log p(r_0) + \sum_{i=1..n} -\log p(z_i|r_i) - \log p(r_i|r_{i-1})$$

Replacing Viterbi algorithm

- **Viterbi**

Maximize:

$$p(R,Z) = p(z_0|r_0)p(r_0)\prod_{i=1..n} p(z_i|r_i)p(r_i|r_{i-1})$$

- **bidirectional Dijkstra‘s algorithm**

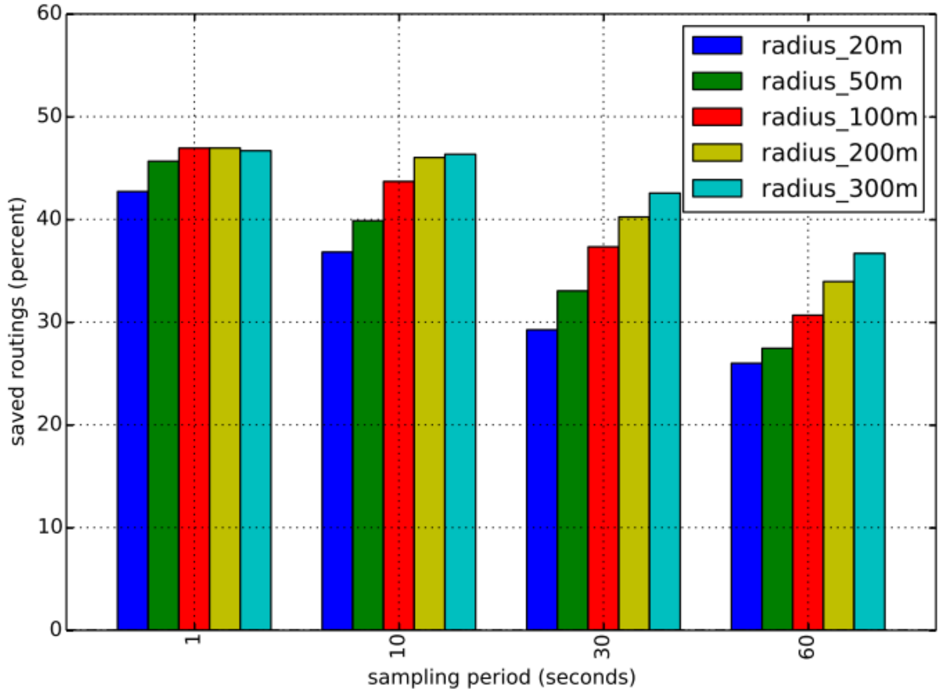
Minimize:

$$C(R,Z) = \sum_{i=0..n} c_{node}(r_i, z_i) + c_{edge}(r_i, r_{i-1})$$

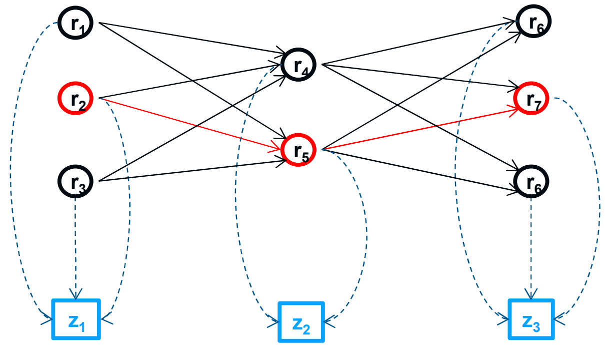
$$-\log p(R,Z) = -\log p(z_0|r_0) - \log p(r_0) + \sum_{i=1..n} -\log p(z_i|r_i) - \log p(r_i|r_{i-1})$$

Experimental results

- Saved routings:



Conclusion and Outlook



- transforming the maximum likelihood problem into a **minimum cost path problem** and **replacing Viterbi algorithm with bidirectional Dijkstra's** algorithm significantly reduces the number of computationally expensive shortest-path routings!

- **savings increase** when the mapmatching algorithm has to account for **greater uncertainty / noise**

- this approach can be **combined with previous approaches** in the literature (multi-threading and optimized shortest-path routing) to further improve performance

- search **algorithms other than bidirectional Dijkstra's algorithm** – e.g. A*-search – have potential for further improvements.
  - required: heuristic for estimating cost from currently visited node to target node (e.g. based on great circle distance?)

Fast HMM Map-Matching for Sparse and Noisy Trajectories

Hannes Koller, Peter Widhalm,
Melitta Dragaschnig, Anita Graser

PETER WIDHALM
Mobility Department
Dynamic Transportation Systems

T +43(0) 50550-6655 | F +43(0) 50550-6439
peter.widhalm@ait.ac.at | http://www.ait.ac.at

HANNES KOLLER
Mobility Department
Dynamic Transportation Systems

T +43(0) 50550-6674 | F +43(0) 50550-6439
hannes.koller@ait.ac.at | http://www.ait.ac.at