Chapter 4

Experimentation and results

4.1 Experiment setup

To test the effectiveness and efficiency of the map-matching algorithm, we needed to be able to test it in a controlled environment with representative data sets as well as a true path for every trajectory to evaluate the accuracy. Since actual data containing true paths for vehicle trajectories is hard to come by and may be fraught with data anomalies, it was decided to generate data sets to test in controlled experiments.

Two networks were used for generating experiments. One was an elementary grid network created using MATSim and the other was the actual road network of the City of Cape Town, in the Western Cape of South Africa. For both networks, random paths and corresponding random global positioning system (GPS) points at different frequencies were generated.

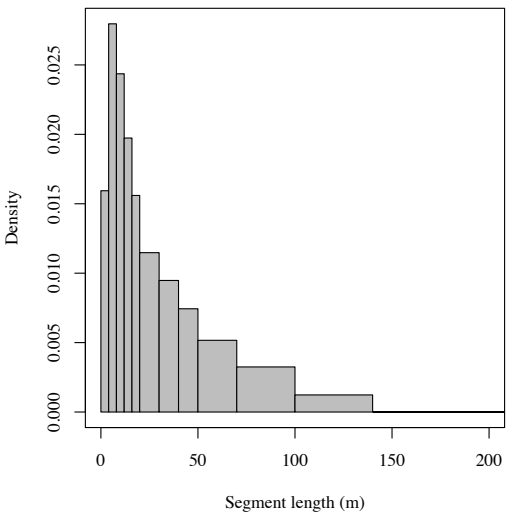
This section aims to discuss setting up the experimental framework for testing the map-matching algorithm.

4.1.1 Analysis of existing road networks

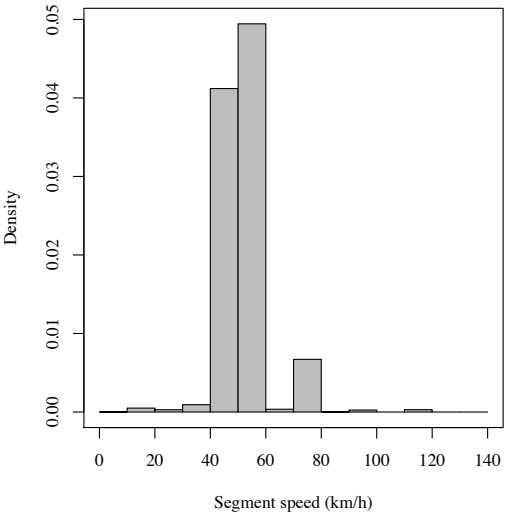
To create experimental grid networks that are comparable to existing road networks, some exploratory data analyses were done to assess network characteristics, especially segment length and speed. The results from the analysis also set the boundary conditions for the conclusions drawn from the algorithm efficiency and effectiveness analysis on the simple grid and real road network experiments.

As seen in Figure 4.1, the City of Cape Town's network comprises of roads of various lengths in the network with an average segment length of 44 m and a median of 27 m. There were some outliers in the data, which have been removed from this figure to avoid impeding the readability. The third quartile is 59 m but the maximum length is 7 949 m. There where fewer than 1% of segments that had a length of less than 1 m. Figure 4.1b illustrates the distribution of segments based on free speed. Since the city comprises mostly residential areas, the segment speed restrictions were found to be predominantly between 40 km/h and 60 km/h with a few 80 km/h segments, which are the main speed restriction categories imposed in residential areas in South Africa. Very few highway road segments of 100-120 km/h were found in the dataset. This is a limiting factor when drawing assumptions for the application of the algorithm in networks that contain highways, and suggests further experiments on such networks.

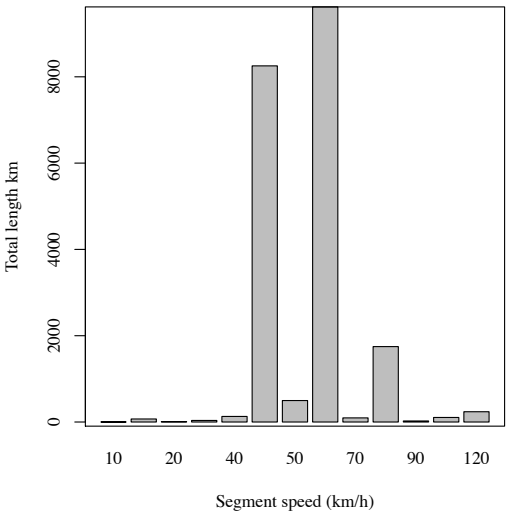
To ensure that the data was not skewed by residential segments at the lower speeds potentially being shorter and thus at a higher frequency, but lower overall distance, a



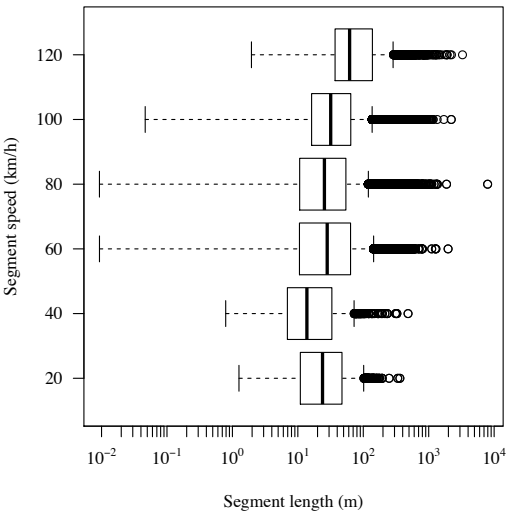
(a) Segment length distribution



(b) Segment free speed distribution



(c) Sum of segment length per free speed



(d) Distribution of segment length to free speed

Figure 4.1: Analysis of Cape Town full network road segment

further analysis was done by adding the length of all segments in a specific speed group. Figure 4.1c shows that summing the length of links into speed groups has the same distribution although there is an increase in the 100-120 km/h range, which was almost non-existent in the previous analysis. Thus even though the number of segments in the 100-120 km/h range, was considerably fewer than any other segment free speed groups, their length was on average more than the lower speed categories. To ensure that there was also no significant relationship between segment length and segment free speed, their relationship was analyzed in Figure 4.1d. The results showed that all segments regardless of their length, had about the same relationship of speed versus length. If the opposite was true, one would have expected to see a greater variance in the distribution, and probably more outliers, resulting in a more dispersed distribution.

The simple grid network needs to be representative of the real-world road network to infer the accuracy and efficiency from the experiments to the real-world data. The results from the Cape Town network analysis were used to parameterise the simple grid network. Given that the simple grid network was purely a theoretical exercise, and subsequent experiments will be conducted on a real-road network, it was decided to keep the grid network very simple with constant speed and length attributes for every segment. This removed the number of factors to consider when drawing conclusions about the influence of key algorithm parameters and network characteristics. The simple grid network was set up to contain segments lengths of 27 m and speeds of 60 km/h.

4.1.2 Methodology for setup of experimental road network

The experimental network was a very simple grid network which was constructed by creating a specified number of horizontal and vertical nodes in a grid fashion. Links were then generated to and from neighbouring nodes and assigned a fixed free speed and length. The algorithm's final output is a network object that can be stored as an XML file to be used in Multi-Agent Transport Simulation (MATSim) as well as viewed in VIA, a specialised software package for rendering MATSim networks and simulations. Figure 4.2 illustrates such an experiment as visualised in VIA.

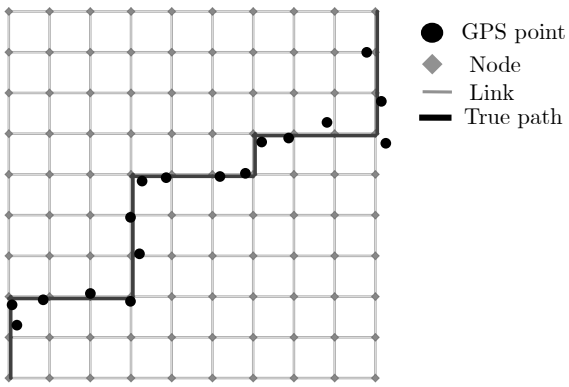


Figure 4.2: Example of simple grid network, true path and GPS trajectory

It is proposed that further experiments using a simple grid network can be conducted using varying speeds, lengths as well as possibly adding links that simulate highways, service roads, and other abnormal road network elements. Given the ubiquitousness of the actual road network, creating experimental networks from scratch will only be useful

to test very specific network designs in a controlled fashion. For this study, we will be testing the algorithms capability on a real-world network, the City of Cape Town, with simulated trajectories and with varying GPS sampling rates.

4.1.2.1 Methodology for generating an experimental true path and GPS trajectory

In the artefacts, two algorithms were used for generating a true path on the networks. One is a systematic approach of moving from one segment to the next starting from the first node to the last node of the network, and the other, a more flexible and sophisticated approach that uses the shortest path between two or more randomly chosen points within a network. The former was used on the simple grid network and the latter for generating paths on the real-world network.

Simple grid network approach: For the simple grid network the true path is generated by starting at the first node in the network, bottom left-hand corner, and working up to the last node, top right-hand corner, as can be seen in Figure 4.2. Starting at the first node the path-generating algorithm randomly chooses an outgoing link connected to this node and checks whether the angle between the current node, the chosen link's destination node and the last node of the network is greater than $\pi/2$. If this is true, the link is chosen and saved; if not, the next alternative link is chosen randomly. If no alternative link is available, the link with the greatest angle from those evaluated is chosen. Links that would lead to a node that had already been traversed in the process of path generation are excluded from the initial list of available links being evaluated. The process then continues from the destination node of the chosen link until the destination node is the last node in the network or there are no alternative links available because they had either been traversed previously or they lead to a node that had been traversed previously. The final output is a network object than can be stored as an XML file.

In a grid configuration with equal lengths, there is always a path generated from the starting node to the final node, but if the algorithm is run on a complex road network, similar to real-world road networks, with dead-ends and roundabouts, the algorithm regularly ends or runs in circles before the final node is reached. To create more realistic true paths on real-world networks, a different approach to simple grid networks are used when generating trajectories.

Real-world network approach: For the real-world network, the path generation uses the Dijkstra method to find the shortest path between two random nodes within the network. To create more complex paths, the algorithm can do this action a number of times and combine the paths by using the end node of the previous step as the starting node for the next step.

4.1.2.2 Generating GPS point

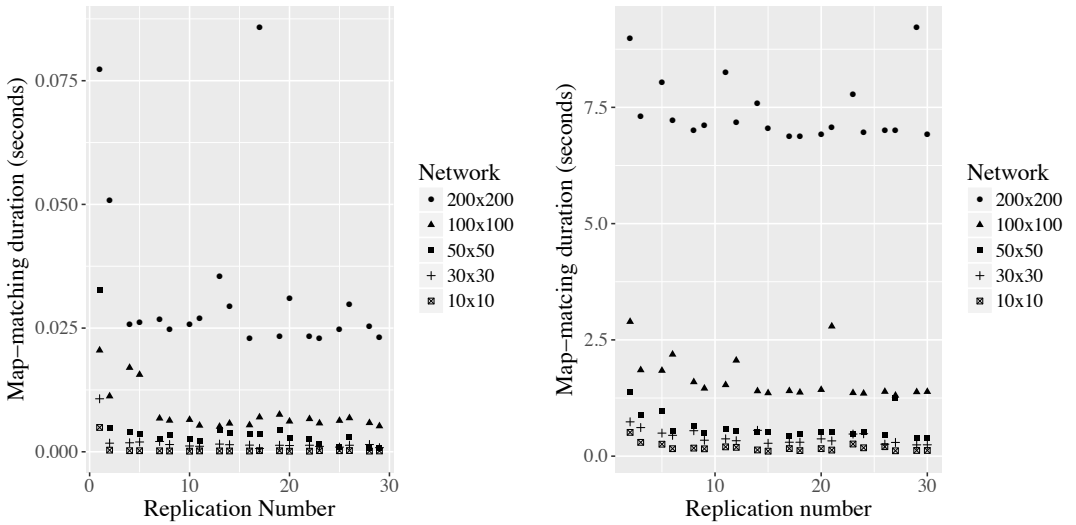
Once the path has been generated a second algorithm will take the path and plot random GPS points along the path based on a specified trajectory sampling rate and GPS error. The GPS trajectory-generating algorithm starts at the first node in the path and moves a distance on the path based on the free speed of the path and the sampling rate. A simulated GPS point is created by sampling a single distance from the distribution $N(\mu = 0, \sigma^2)$, and takings its absolute value to represent the distance from the actual point on the link

to the GPS point. The direction of the point is based on an angle sampled from a uniform distribution in the range $[0, 2\pi]$. Using Pythagoras' theorem the coordinates for the GPS point is calculated and stored. See Figure 4.2 for the final output once a grid network has been generated and a true path and GPS points generated.

4.2 Experiment replication

During the experimentation and investigation phase, it was found that the first iteration of the algorithm performs considerably slower than the subsequent runs of the algorithm. Figure 4.3 shows the duration of independent experiments with multiple replications of exactly the same setup parameters. It is evident that the first instances of any experiment set have a longer execution time than the rest. This is true for both the quadtree creation as well as the map-matching algorithm. Interestingly, the difference between the initial run and than the rest of the replications seems to increase with the increase in network size suggesting the additional delay is a function of the network and not a fixed time applicable to all experiments.

What is peculiar, however, is the fact that some unknown event is causing erratic durations in the execution time of the map-matching algorithm and, as expected, the outcome of this is more profound in the bigger networks than in the smaller ones, suggesting a relationship with network size and additional duration.



(a) Quadtree creation versus replication number (b) Map.matching duration versus replication number

Figure 4.3: Efficiency analysis on number of replications

The cause of this was not fully investigated, as it was not the primary focus of the current study. The possible causes of this could include implementation inefficiencies as well as possible class loading or dynamic linking of native methods inside the Java Virtual Machine. The abnormal occurrences later on in the replications could also have been caused by the garbage collector, which is Java's automatic memory management process that involves removing redundant classes and objects from memory. This is likely why there was a bigger influence on the larger network sizes where there is a higher probability of the garbage collector running due to a larger number of classes and objects created to

facilitate the map-matching process. Unfortunately, in Java there is no external control over the garbage collector. These inefficiencies present opportunity for further analysis.

Since the initial runs of the algorithm might skew the results, the first run of every experiment is removed from the analysis by letting the algorithm run one iteration before recording all the duration stats. All experiments where tun for 20 replications of the same experimental setup unless otherwise stated.

4.3 Network size

This section aims to identify the influence of changes in network size on the map-matching algorithms efficiency and will indicate if preprocessing of networks can yield any benefit.

4.3.1 Network size increase with a fixed path

This experiment aimed to identify the influence of the overall network size if the GPS trajectory stays constant. This had to indicate whether there is a need to reduce the network object used during map-matching so that the network only includes segments that are likely to be evaluated as part of the map-matching algorithm. This might increase the efficiency of the algorithm in networks such as the City of Cape Town where the overall network contains 467 748 links but an object might traverse only a very small portion of the network.

For this experiment a true path was generated on a small 10×10 network and kept constant while the network size was increased to 30×30, 50×50, 100×100 and 200×200. In Figure 4.4 the true path is indicated by the dark black line while the different rectangles indicate the different road networks generated.

Segment lengths and speed were kept at a constant 27 m and 16.667 m/s (60 km/h) respectively, as defined in the experiment setup section (see 4.1). All the other parameters were kept constant for each replication and sub-experiment, namely the GPS period was 1 s and number of closest links to analyse was set to 8 in the-matching algorithm.

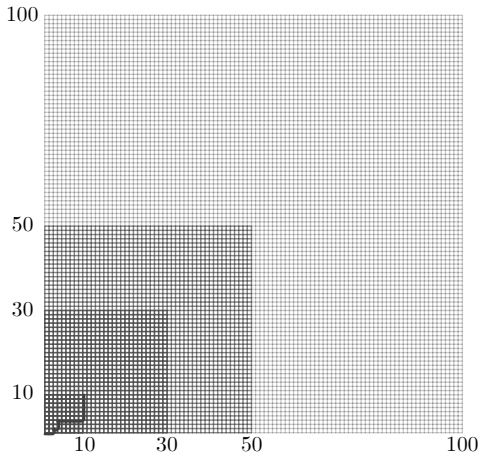


Figure 4.4: Increase in network size grid analysis

As can be seen in Figure 4.5, there is very little difference in the execution time of the map-matching algorithm between grid networks 10×10 to 50×50; however, when

moving to the bigger networks of 100×100 and 200×200, there seems to be an increase in computational overhead.

This happens because, as the network becomes bigger the possible routes to assess for the temporal part of the algorithm increases. The temporal part of the algorithm uses the Dijkstra method, which is the most computationally expensive part of the algorithm.

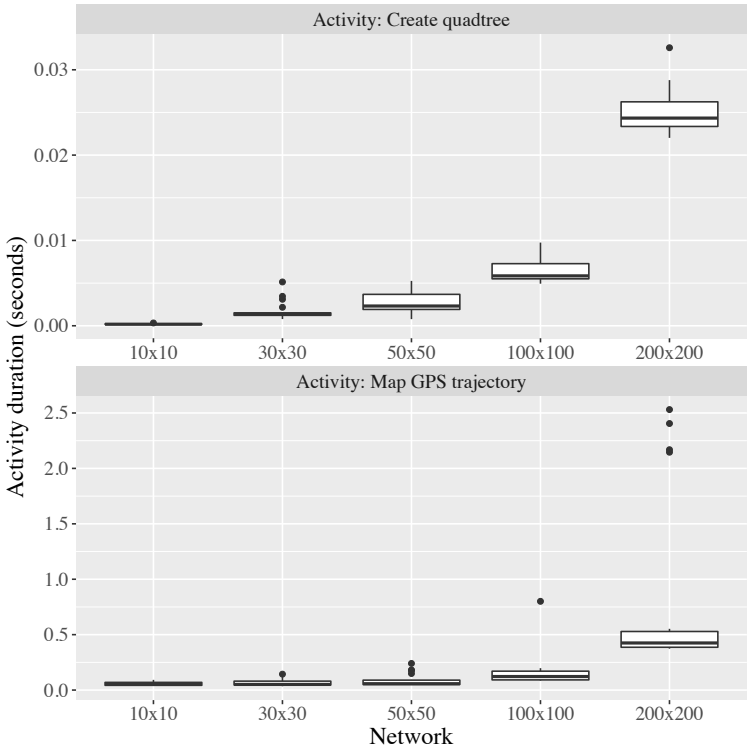


Figure 4.5: Increase in network size efficiency analysis

As expected, the creation of the quadtree was directly correlated to the grid size, the correlation of which was further analysed, as described in section 4.3.2. The creation of the quadtree takes noticeable computation time but is only done as part of the initial setup of the map-matching algorithm. The same quadtree is used for all subsequent matches on the same network and, as such, the duration of the map-matching process is far more important, as it will be used for the mapping of every trajectory.

The map-matching function was also influenced by the grid size, which suggests that removing unnecessary parts of the network will yield a reduction in execution time of the algorithm. Further analysis yielded that the most time-consuming step in map-matching is creating the graph of candidate links and more specifically the weighting of the links in the graph. The most time-consuming activity during the weight assignment of each link is caused by the Dijkstra method. The Dijkstra method is used when finding the shortest path between two subsequent GPS points for which candidate links are not connected at a common node.

4.3.2 Network size on creation of quadtree

The creation of the quadtree is directly correlated to the size of the network and was potentially a time-consuming activity. This section provides a brief explanation of how the quadtree is generated and the analysis done to quantify the outcome in relation to network size.