Implementation of a Fuzzy Logic Based Map Matching Algorithm in R

Bachelor Thesis
Nikolai Gorte
375159
n_gort01@uni-muenster.de
University of Münster

28.02.2014

Supervisors
Prof. Dr. Edzer Pebesma
Institute for Geoinformatics
University of Münster

Dr. Christoph Stasch
Institute for Geoinformatics
University of Münster

Abstract

The Global Positioning System (GPS) is widely used and is a major positioning technology for land vehicle navigation. However it is not 100% accurate, which is a problem for any kind of navigation system. There are several factors that contribute to positioning errors, e.g., satellite related errors, propagation related errors, receiver related errors, GPS signal masking or blockage and satellite geometric contribution to position error (Quddus, 2006). This is where `map matching` comes in.

Map matching is the process of matching GPS trajectories to a digital road network. This is done by map matching algorithms. In Quddus (2006) several of the existing map matching algorithms are discussed and three improved map matching algorithms are introduced. One of the more successful algorithms is the fuzzy logic map matching algorithm, whose implementation is the main part of the bachelor thesis.

The above mentioned map matching algorithm is implemented in `R` (R Core Team, 2013) and therefore it is open source. The testing of the algorithm is done using field data acquired from the `enviroCar`[1] project.

The result of this bachelor thesis is a documentation and an `R package` providing functions which allow the user to match their GPS trajectories to a digital road network using the fuzzy logic map matching algorithm and also allow the customization of the parameters used in the membership functions in the fuzzification process.

[1] `https://www.enviroCar.org/`

Contents

List of Figures

1 Introduction

1.1 Background

Since its launch in 1978 GPS has become an important global utility and an integral part of global security, economy and transportation safety (DoD, 2008). Originally developed for military use, GPS nowadays also provides positioning, navigation and timing services for civil and commercial users around the world. In this research especially its importance for transportation technologies, such as personal navigation assistance, matters.

Nowadays a wide variety of `Advanced Transport Telematics Systems` (ATTS) which depend on GPS are available. ATTS can be described as

> the integrated application of advanced sensors, computers, electronics, navigation, and communication technologies to vehicles and roadways to increase safety, reduce congestion, enhance mobility, minimize environmental impact, increase energy efficiency, and promote economic productivity for a healthier economy(Quddus, 2006, p. 2).

Basically this includes all kinds of advanced personal navigation assistance systems and systems for monitoring and tracking vehicle movement. The data collected from such systems helps to understand the behavior of travelers and shows potential consequences to the environment and the transport system (Quddus *et al.*, 2003).

All these systems build on two essential components which are a device to obtain the position of a vehicle, e.g. GPS, and a digital road map for reference of the vehicle location. Ideally both of these components are 100% accurate and the vehicle location coincides with the digital road network. Unfortunately GPS and also digital road networks are affected by errors. GPS is affected by systematic errors, random errors and errors in the digital road network arise during the creation and digitalization of maps (Quddus, 2006). This means that it is very unlikely that the locations obtained from GPS and the digital road network coincide and additional work, that is called map matching, is needed to reconcile them.

1.2 Problem statement

In White *et al.* (2000) the map matching problem is described as follows. Assume a vehicle is moving along a finite system of streets ,$\overline{\mathcal{N}}$, we do not know. Instead we have a network representation, $\mathcal{N}$, of the system in $\mathbb{R}^2$. For simplicity it is assumed that there is a one-to-one correspondence between the streets in $\overline{\mathcal{N}}$ and the arcs in $\mathcal{N}$

$\mathcal{N}$ consists of piece-wise linear arcs representing single roads. Each arc, $A \in \mathcal{N}$, can be described by a finite sequence of points, $(A_0, A_1, ..., A_{n_A})$. These points are the endpoints of the line segments arc A consists of. The points A_0 and A_{n_A} are called nodes. These are the endpoints of an arc and often points on which it is possible to move to another arc, hence they correspond to dead-ends or junctions in the street system.

Additionally the GPS device provides us with an estimate of the vehicle's location at time t. The actual location of the vehicle at time t is denoted by $\overline{\mathcal{P}_t}$ and the estimated location is $\mathcal{P}_t$.

The goal is to find the road, $\overline{A} \in \overline{\mathcal{N}}$, that corresponds to the vehicle's actual location, $\overline{\mathcal{P}_t}$, by matching the estimated location, $\mathcal{P}_t$, with an arc $A \in \mathcal{N}$.

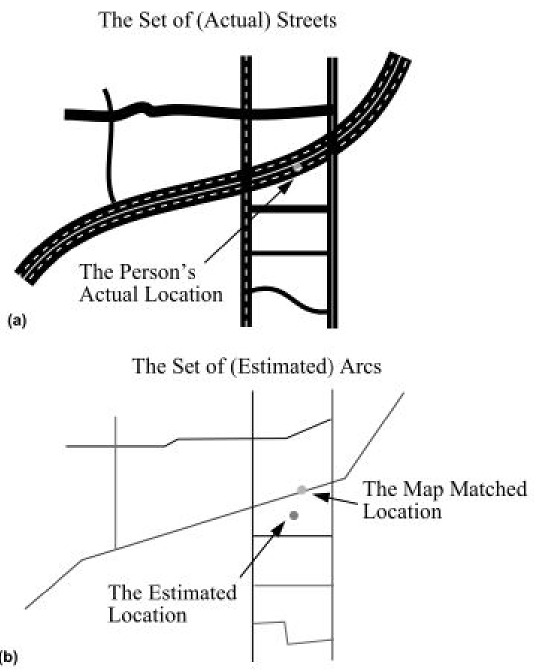
The Set of (Actual) Streets
The Person's Actual Location
(a)
The Set of (Estimated) Arcs
The Map Matched Location
The Estimated Location
(b)

Figure 1: The Map Matching Problem (White *et al.*, 2000, p. 94)

1.3 Aims

Of the several existing map matching algorithms discussed in Quddus (2006) the fuzzy logic map matching algorithm is the most successful in terms of identifying correct links in the digital road network, as can be seen in Quddus (2006, p. 215). Hence, the purpose of this research is the implementation of the fuzzy logic based map matching algorithm in the `R` programming language.

The main motivation for this is the enviroCar[2] project, a collaborative project of 52north[3] and the Institute for Geoinformatics at University of Münster[4]. EnviroCar is an open platform for citizen science that allows you to investigate the environmental effects of your driving. Using an Android[5] app users can submit their recorded trajectories including data such as fuel consumption, CO2, noise emission etc. and visualize them on a map. The algorithm implemented in this research can be used to match these trajectories to the digital road network.

The result of this research is an `R` (R Core Team, 2013) package called `fuzzyMM` that can be used to reconcile inaccurate vehicle locations obtained from GPS with a digital road network.

2 A Fuzzy Logic Map Matching Algorithm

This section shortly introduces Sugeno's fuzzy inference system and describes the fuzzy logic map matching algorithm that builds upon it and whose implementation is the main goal of this research. A more detailed description of the algorithm and the fuzzy inference system is found in Quddus (2006).

2.1 Sugeno's Fuzzy Inference System

Fuzzy logic was introduced by Zadeh in the 1960's. He worked in the field of control engineering and developed the fuzzy theory in order to deal with problems where knowledge of experienced human operators, expressed in vague, linguistic terms is needed to control certain processes (Quddus, 2006). In order to represent the meaning of linguistic terms "the concept of grades of membership or the concept of possibility values of membership was introduced" (Quddus, 2006, p. 143).

The point is basically to map an input space to an output space using a list of

[2]`https://www.envirocar.org/`
[3]`http://52north.org/`
[4]`http://www.uni-muenster.de/Geoinformatics/en/`
[5]`http://www.android.com/`

if-then-rules (MathWorks, 2013a). The whole system doing this is called Fuzzy Inference System (FIS) and is shown in Figure 2.

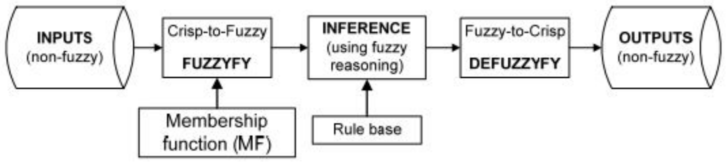

Figure 2: Fuzzy Inference System (FIS) (Quddus, 2006, p. 144)

In Quddus (2006) the functionality of a FIS is described as follows.
Assume:

- If the speed of the vehicle is high and the travel time is low then the traffic congestion on the link is low

is a knowledge-based fuzzy rule. In this example `speed` and `travel time` are the `input variables` for the FIS and `high` and `low` are so called `input fuzzy subsets`. Hence, the `traffic congestion` is the `output variable` and `low` is the `output fuzzy subset`. In a process called `fuzzification` the input variables now are assigned to membership values between 0 and 1 using membership functions (MF). After the fuzzification process the output of the if-then-rules is calculated by applying a minimum method or a product method to the fuzzified inputs and then using an output MF. The outputs of each rule are then aggregated and afterwards deffuzified using methods such as centroid calculation, weighted average, etc. The final output is a single number called crisp.

Sugeno's FIS is one of the main components of the fuzzy logic map matching algorithm because it is used to calculate the possibility of matching a position obtained from GPS to a certain link in the digital road network. However, it slightly differs from the FIS described above. While the fuzzy rules in the FIS described above look as follows:

$$If\ x\ is\ A\ and\ y\ is\ B\ then\ z\ is\ C$$

the rules in Sugeno's FIS have the following form:

$$If\ x\ is\ A\ and\ y\ is\ B\ then\ Z = f(x, y)$$

with $Z = f(x, y)$ being a linear function, $Z = ax + by + c$ or a constant in case $a = b = 0$ (MathWorks, 2013b). Figure 3 shows how Sugeno's FIS works.

In Figure 3 the input variables for Sugeno's FIS are $PQ = 15m$ and $\Delta\theta = 78°$. After the fuzzification, using trapezoidal MF, a minimum method is applied to

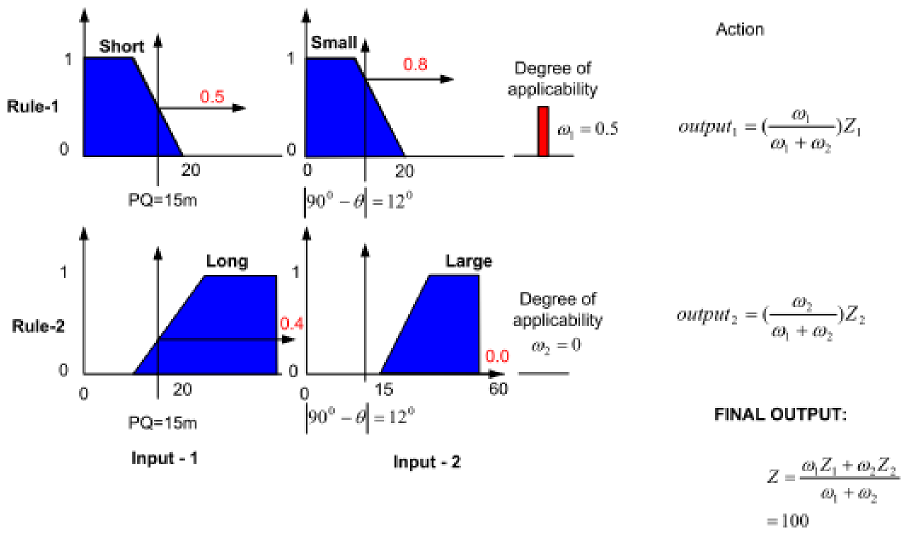

Figure 3: Sugeno's Fuzzy Inference System (Quddus, 2006, p. 151)

the outputs of the MF to obtain the degree of applicability, ω, for each rule. Z is simply a constant and the output of each rule is:

$$output_j = (\frac{\omega_j}{\sum_{i=1}^{N} \omega_i})Z_j, \quad N = number\ of\ rules$$

The final output of the FIS is the weighted average of all rule outputs:

$$Z = \frac{\sum_{i=1}^{N} \omega_i Z_i}{\sum_{i=1}^{N} \omega_i}.$$

2.2 Map Matching Process

The map matching process of the fuzzy logic map matching algorithm is split into two parts. The first and most challenging part is the identification of the correct link in the digital road network, which is done in the initial map matching process (IMP) and in the subsequent map matching process (SMP). The second part is the determination of the vehicle location on a link (Quddus, 2006). All the processes are described below.

2.2.1 Initial map matching process (IMP)

In Quddus (2006) the IMP is described as follows. It is the first and very important part of the map matching process because it is responsible for the identification of the initial link. An error in this process will probably lead to incorrect subsequent matchings.

The IMP relies on an elliptical or rectangular error region formed around the GPS position that needs to be matched to the road network. This error region is derived using the variance-covariance matrix (VCV) associated with the errors in the computed GPS position. Unfortunately the calculation of the VCV requires data the trajectories from the enviroCar project do not include (e.g. range between the satellite and the receiver, the topocentric distance between the satellite and the receiver etc.). Because of this an error rectangle with fixed size will be used in this research. More information about this are given in the implementation section.

After the error rectangle is formed, all the links within the rectangle are considered to be candidate links and the FIS for the IMP is used to identify the correct link among the candidate links. The input variables of this FIS are the speed of the vehicle, v (m/sec), the horizontal dilution of precision (HDOP), the perpendicular distance to the link, PD (m) and the heading error, HE (degrees) which is the absolute difference between the direction of the vehicle and the direction of the link. The output of this FIS, L1, is the probability of matching the position fix to a candidate link. In the fuzzification process, sigmoidal MF are used as shown in Figure 4. The constants used to calculate the output, L1, are low , $(Z1) = 10$, $average$, $(Z2) = 50$ and $high$, $(Z3) = 100$.

The fuzzy rules used in this FIS are:

- If (v is high) and (HE is small) then (L1 is average)
- If (v is high) and (HE is large) then (L1 is low)
- If (HDOP is good) and (PD is short) then (L1 is average)
- If (HDOP is good) and (PD is long) then (L1 is low)
- If (HE is small) and (PD is short) then (L1 is high)
- If (HE is large) and (PD is long) then (L1 is low)

A link is chosen as initial link if the FIS identifies it as the correct link for the first few position fixes.

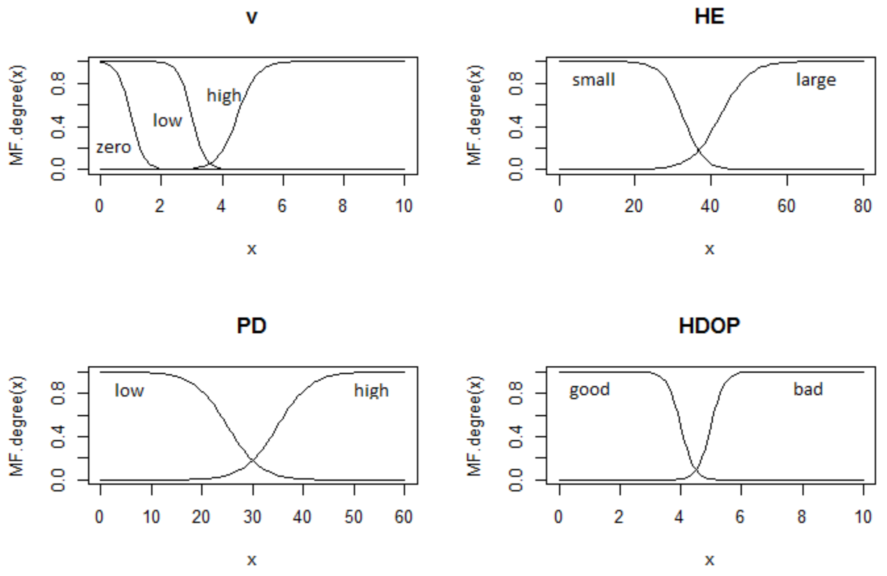

Figure 4: Fuzzification of the inputs during the IMP

2.2.2 The subsequent map matching process (SMP)

The SMP starts after the successful execution of the IMP and is responsible for the matching of the subsequent positions. It is split into two parts:

1. SMP along a link (SMP-1) checks if the vehicle has already crossed a junction or is about to cross one and matches subsequent positions to the link identified by the IMP if that is not the case (Quddus, 2006).
2. SMP at a junction (SMP-2) identifies a new link among the candidate links if the vehicle has crossed a junction (Quddus, 2006).

Quddus (2006) describes them as follows.

SMP along a link (SMP-1) SMP-1 determines the possibility of matching the subsequent position fix to the previously selected link. So there is no need to create an error rectangle and look for candidate links.

The inputs needed for the FIS are the speed of the vehicle, v (m/sec), the HDOP, Δd (m), the heading increment, HI, α (degree) and β (degree), where

- Δd: the difference between the distance from the last matched position to the downstream junction and the distance traveled by the vehicle since the last position fix
- HI: the absolute difference between the direction of the vehicle at the current position and the last position
- α and β: location of the current position fix, relative to the link, seen from the last matched position and from the downstream junction respectively.

Additionally it is possible to use the gyro-rate reading, $\Delta\theta$, but this is data not recorded by enviroCar and probably not accessible with most smartphones. Therefor we will not use the gyro-rate reading as input to the FIS.
The fuzzy rules (without gyro-rate) for this FIS are:

- If (α is below 90) and (β is below 90 $^\circ$) then (L2 is high)
- If (Δ d is positive) and (α is above 90 $^\circ$) then (L2 is low)
- If (Δ d is positive) and (β is above 90 $^\circ$) then (L2 is low)
- If (HI is small) and (α is below 90 $^\circ$) and (β is below 90 $^\circ$) then (L2 is high)
- If (HI is small) and (Δd is positive) and (α is above 90 $^\circ$) then (L2 is low)
- If (HI is small) and (Δd is positive) and (β is above 90 $^\circ$) then (L2 is low)
- If (HI is large) and (α is below 90 $^\circ$) and (β is below 90 $^\circ$) then (L2 is low)
- If (HDOP is good) and (v is zero) then (L2 is high)
- If (HDOP is good) and (Δd is negative) then (L2 is average)
- If (HDOP is good) and (Δd is positive) then (L2 is low)
- If (v is high) and (HI is small) then (L2 is average)
- If (HDOP is good) and (v is high) and (HI is 180 $^\circ$) then (L2 is high)

The output of the FIS, L2, is a threshold value used to determine if the position fix should be matched to the previously selected link. According to Quddus (2006) a value of 60 is adequate to assume the vehicle is still on the same link.

SMP at a junction (SMP-2) As described in Quddus (2006) the SMP-2 is used when the vehicle is either about to cross or has just crossed a junction. SMP-2 basically works the same way IMP does but with two additional input variables and four additional rules.

The new variables are link connectivity and distance error (m). The link connectivity simply tells if a candidate link is directly connected to the previous selected link or not and the distance error is the difference between the distance traveled from the last position (calculated from the speed of the vehicle) and the shortest path from the last matched position to the current position along the links.

The new fuzzy rules involving these variables are:

- If (The connectivity with the previous link is low) then (The L3 is low)
- If (The connectivity with the previous link is high) then (The L3 is high)
- If (The distance error is low) then (The L3 is high)
- If (The distance error is high) then (The L3 is low)

The output of the FIS, L3, is the likelihood of a candidate link to be the correct link.

2.2.3 Determination of the vehicle location on a link

After the right link among the candidate links is found the vehicle location on the link is needed. For determining the vehicle location on a link Quddus (2006) describes an optimal estimation technique. Unfortunately this also involves the calculation of the VCV, which cannot be done because of the lack of the necessary data in the enviroCar trajectories. So in this research the vehicle location on a link will be determined using an orthogonal projection of the position fix on the link. This is not an optimal solution but a sufficient one since the main challenge is to find the correct links for the GPS positions.

3 Implementation

After the description of the fuzzy logic map matching algorithm this section now describes its implementation in `R` (R Core Team, 2013). It talks about the needed inputs, creation of the digital road network and shows how the implementation of the different parts of the algorithm such as IMP, SMP and FIS was handled. It also talks about problems or challenging aspects in each part of the implementation.

3.1 Structure

We have seen that the algorithm itself consists of several different parts and so does the implementation. The first steps were to think about the inputs needed and acquisition of map data that fulfills the requirements to do the map matching. The next step was to find a way to implement the three different FIS that are a core element of the IMP and the SMP. After the implementation of the FIS, the IMP and the SMP were implemented and everything was put together.

3.2 Input

The input to the fuzzy logic map matching algorithm is a SpatialPointsDataFrame as defined in the `sp` package (Bivand *et al.*, 2013). It contains the coordinates of the GPS trajectory that needs to be matched to a digital road network. Additionally it must include following data to be processed by the algorithm:

- GPS.Speed in km/h
- GPS.Bearing
- GPS.HDOP
- time as "POSIXct" or "POSIXlt"

A projection also must be specified. Missing values in the data are replaced with zeros and can lead to incorrect matchings of the corresponding points and of the subsequent points.

3.3 Digital road network

The digital road network is one crucial component needed in the map matching process and not fairly hard to obtain because there is no implementation of a class in `R` that fulfills all the needs required by the algorithm. Several `R` packages exist that allow the import of map data from different sources but mostly in form of raster data which is not suitable for this algorithm.

As seen in Chapter 2 we need a digital road network that allows us to do shortest path calculations, get information about link connectivity, get the length of the links etc. To achieve this we use the `osmar` (Eugster & Schlesinger, 2010), `igraph` (Csardi & Nepusz, 2006) and `sp` (Bivand *et al.*, 2013) packages to implement a structure usable in the map matching process.

First of all the road data is obtained from OpenStreetMap[6] (OSM) using the Overpass API[7]. The advantage of this API is that it allows to get data from OSM by search criteria such as location, tag, properties etc. In `R` a request to obtain the data looks as follows:

```
> url <- paste0("http://www.overpass-api.de/api/xapi?way
+               [bbox=",x1,",",y1,",",x2,",",y2,"][highway=*]")
> response <- getURL(url, .encoding = "UTF-8")
```

This request gets all the features inside the bounding box that are tagged as highways, which means we download almost no unnecessary data. The coordinates for the bounding box are obtained from the bounding box of the input file, the SpatialPointsDataFrame.

Then we transform the data into an `osmar` object. This allows us to further filter the data and remove features we do not need (e.g. cycle ways, foot ways, bridleway etc.) using the tags of the OSM features. This needs to be done to prevent errors such as matching vehicle positions to cycle ways running close to the roads.

```
> resp <- xmlParse(response)
> roads <- as_osmar(resp)
> # Get ID's of all streets used by cars
> id <- find(roads,
+            way(tags(k == "highway" & !(v %in% c("cycleway",
+                                           "footway", "bridleway",
+                                           "steps", "path")))))
> roads <- subset(roads, ids = find_down(roads, way(id)))
```

The next step is to create an `igraph` object from the the `osmar` object. This allows us to calculate the shortest path between nodes and the length of the shortest path. This is fairly easy because the length of the line segments in the `osmar` object, given in meters, is the weight of the edges in the graph. This is useful in SMP-2 where shortest path calculation is needed.

```
> graph <- as_igraph(roads)
> graph <- as.undirected(graph, mode = "each")
```

Now we can transform the `osmar` object into a `SpatialLines` object where each line is split into its segments.

[6] `http://www.openstreetmap.org`
[7] `http://overpass-api.de/`

```
> roads <- as_sp(roads, "lines")
> roads <- lines2segments(roads)
```

The edges of the graph now correspond to the line segments in the `SpatialLines` object. This combination allows us to find the roads inside the error rectangle and then the corresponding nodes in the graph, which makes it easy to obtain the needed shortest paths and connections between edges. Now we can combine the graph and the SpatialLines into a DigitalRoadNetwork.

```
> setClass("DigitalRoadNetwork",
+          representation(sl = "SpatialLines", g = "igraph"),
+          validity = function(object) {
+            stopifnot(length(object@sl) == length(E(object@g)))
+          })
> roads <- new("SpatialNetwork", sl = roads, g = graph)
```

This DigitalRoadNetwork allows us to do the map matching.

It should be noted that this process can be very time consuming if data of very large areas or areas with high road density needs to be downloaded and processed.

3.4 Implementing the FIS

As already mentioned in Chapter 2 the FIS is a core element of the algorithm. In total three different FIS are needed, one for the IMP, one for the SMP-1 and one for the SMP-2. The implementation of the FIS was done using the `frbs` package, which allows the creation of different FIS, such as Mamdani's FIS, Sugeno's FIS etc. (Riza *et al.*, 2013). The implementation will be shown using the FIS for the IMP as an example.

The first steps in this process are very straightforward. First thing we need to do is define the model of the FIS. The FIS we are using is Sugeno's FIS and the corresponding model in `frbs` is the Takagi Sugeno Kang (TSK) model.

```
> type.model <- "TSK"
```

Next we set the implication function (minimum function) used to get the degree of applicability of each rule.

```
> type.implication.func <- "MIN"
```

We also have to provide the names of the input variables, the fuzzy subsets of each input variable, the number of fuzzy subsets of each variable and the range of the fuzzy subsets (e.g. speed is high (3-6 m/s)), which is very easy and therefore will not be further explained here.

The fuzzy rules need to be stored in matrix form which requires them to be of the same length. This means each rule must include all of the input variables. Of course not every variable is needed in each rule. In this case the fuzzy subset for the variable becomes "*dont_care*". For example, the fuzzy rules

- If x is A and y is B then Z = f(x,y)
- If x is C then Z = f(x,y)

need to be written as

- If x is A and y is B then Z = f(x,y)
- If x is C and y is *dont_care* then Z = f(x,y)

We define the matrix containing the rules as follows:

```
> r1 <- c("high","and","small","and","dont_care",
+         "and","dont_care", "->")
> r2 <- c("high","and","large", "and", "dont_care",
+         "and", "dont_care", "->")
> # ...
> r5 <- c("dont_care", "and", "small","and","short",
+         "and", "dont_care", "->")
> r6 <- c("dont_care", "and", "large","and","long",
+         "and", "dont_care", "->")
> rule1 <- list(r1, r2, r3, r4, r5, r6)
> rule1 <- do.call(rbind, rule2)
```

It is important to note that the variable names are not mentioned in the definition of the rule and there is also no entry for the then-part of the rules.

The linear functions in the then-part of the rules are defined in a separate matrix, where the number of rows corresponds to the number of rules and the number of columns is the number of variables +1. Since we only use constants, such as $Z1 = 100$, we are only interested in the last column. This allows us to define the matrix with just one column containing the constants for each rule.

```
> ## Define linear functions of TSK
> func.tsk1 <- matrix(c(50, 10, 50, 10, 100, 10),
+                     nrow = 6, byrow = TRUE)
```

The most challenging part in the implementation of the FIS is the definition of the MF. They are defined in a matrix containing the parameters to form the MF. The number of rows is 5 and the number of columns is the number of fuzzy subsets used in the FIS. The number in the first row defines the type of the MF, such as triangle, trapezoid, sigmoid, etc. and the following rows are filled with the parameters of the functions. The problem is that we mostly use sigmoidal MF, as seen in Figure 4. In Riza *et al.* (2013) these are defined as

$$f(x) = \frac{1}{(1 + \mathrm{e}^{-b(x-c)})}$$

where b is the slope at the crossover point c.

So the parameters we need are b and c. Unfortunately these are not given and need to be calculated from the left and right bounds of the fuzzy subsets. To do so we use the function *get_params*:

```
> get_params <- function(l, r, shape = c("s", "z")) {
+   shape <- match.arg(shape)
+   if (shape == "s")
+     y <- c(0.01, 0.5, 0.99)
+   else
+     y <- c(0.99, 0.5, 0.01)
+   x <- c(l, (l + r)/2, r)
+   slope <- ifelse(shape == "s", 1/(r - l),  1/(r - l))
+   data <- list(x = x, y = y)
+   fitModel <- nls(y ~ a/(1 + exp(-b * (x - ((l + r)/2)))),
+                   data, start = c(a = 1, b = slope),
+                   algorithm = "port")
+   # get the coefficients using the coef function
+   params <- coef(fitModel)
+   params[2]
+ }
```

This function takes the left, l, and right, r, bound and the shape of the sigmoidal MF (s = ascending, z = descending) as input and calculates parameter b using the `nls` function. The crossover point, c, is simply $(l + r)/2$.

The default bounds are taken from Quddus (2006). Of course the user can adjust the MF to his needs by changing the range of the fuzzy subsets of each variable. For example, the default range of the fuzzy subset `high` of the variable `speed` is 3 - 6 (m/sec), which can be looked up using *get_var_bounds*(). Using the function *set_var_bounds* allows to change the range of the fuzzy subsets and *update_mf*() updates the MF and the FIS.

The matrix containing the parameters of the MF is then filled using the above implemented *get_params* function.

```
> matrix(c(6, get_params(var_bounds[1, 1], var_bounds[1, 2], "s"),
+          get_mid(var_bounds, 1), NA, NA, #speed_high
+          6, get_params(var_bounds[2, 1], var_bounds[2, 2], "z"),
+          get_mid(var_bounds, 2), NA, NA, #sped_low
+          6, get_params(var_bounds[3, 1], var_bounds[3, 2], "z"),
+          get_mid(var_bounds, 3), NA, NA, #speed_zero
+          # ...
+          6, get_params(var_bounds[8, 1], var_bounds[8, 2], "z"),
+          get_mid(var_bounds, 8), NA, NA, #HDOP_good
+          6, get_params(var_bounds[9, 1], var_bounds[9, 2], "s"),
+          get_mid(var_bounds, 9), NA, NA), #HDOP_bad
+        nrow = 5, byrow = FALSE)
```

The FIS is created by the *frbs.gen* function, taking all of the above as input.

3.5 IMP and SMP

Now that the digital road network and the FIS are prepared, we can implement the IMP and SMP. Since these processes share lots of common elements such as the identification of candidate links and preparing input data for the FIS we will not discuss each process individually. Below some of the more important aspects of these processes are discussed.

3.5.1 Error region

The error region is used in IMP and SMP-2. It is setup on a position fix to get the candidate links for the map matching. As already said in Chapter 2 we cannot use the method described in Quddus (2006) to obtain the size of the error rectangle, so we will use a fixed size based on the horizontal error of GPS. According to DoD (2008, B-37) there is a “99.9% availability of a horizontal accuracy of 38.2m 95% or better any random time at any random location”. This means there is a probability of 95% or better that the measured position lies within a circle with a radius of 38m around the true position. So we create the error region as a `SpatialPolygon` with each side being $2 \cdot 38$m long as seen in Figure 5.

Using the `rgeos` package (Bivand & Rundel, 2013) we can get the ID’s of the links lying within the error region or intersecting the error region. These links are the candidate links used for the map matching.

```
> candidate_links <- data.frame(
+   edge_id = unique(c(which(gIntersects(rec, roads@sl, byid = TRUE)),
+                      which(gContains(rec, roads@sl, byid = TRUE)))))
```

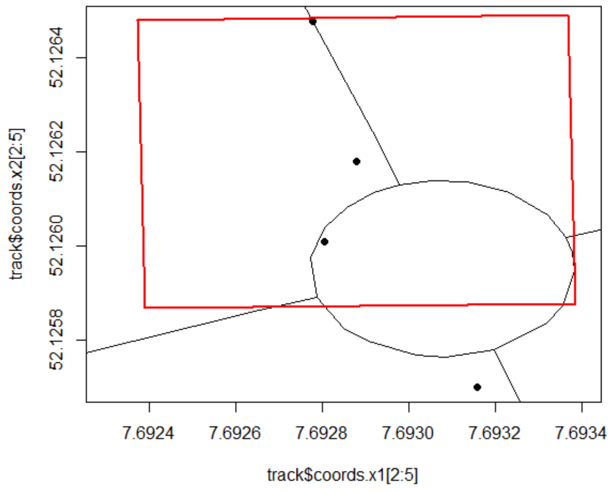

Figure 5: Error region

3.5.2 Bearing and shortest distance to line

To calculate the bearing of the links and the shortest distance of a GPS position to a certain link we use the `geosphere` package (Hijmans, 2014). Bearing and shortest distance to line are both important in the IMP and SMP-2 and are acquired using the *bearing* and *dist2Line* functions. Additionally *dist2Line* does not only calculate the shortest distance to a line but also the nearest point on the line. This is very useful because the nearest point on the line is the point the GPS position is matched to, as described in Chapter 2.

3.5.3 Link connectivity

For the SMP-2 it is important to know which links are connected. This can be checked using the graph of the road network. The igraph class allows to get all edges starting at a specific node which allows to solve the problem easily.

```
> candidate_links$conn <- sapply(candidate_links[,c("edge_id")],
+     function(x) {
+       conn_edges <- E(roads@g)[from(prev_link_end)]
+       if (isTRUE(any(as.vector(conn_edges) == x))) 1 else 0})
```

This simply checks if any of the candidate links is connected to the endpoint of the last selected link.

3.5.4 Shortest path

For the SMP-2 we also need to calculate the shortest path between the last selected link and the candidate links. To do this we use the graph of the street system. Since we do not know which node of the candidate link is the nearest we calculate the shortest path to both nodes and choose the one with the shortest distance.

```
> sp <- as.data.frame(do.call(rbind, lapply(1:nrow(candidate_links),
+       function(x) {
+         spV1 <- shortest.paths(roads@g, prev_link_end,
+                                candidate_links$V1[x])
+         spV2 <- shortest.paths(roads@g, prev_link_end,
+                                candidate_links$V2[x])
+         if (spV1 < spV2) {
+           c(candidate_links$V1[x], spV1)
+         } else {
+           c(candidate_links$V2[x], spV2)}}
+       )))
>   candidate_links$cl_vertex <- as.character(as.vector(sp[,1]))
>   # length of the shortest path (m)
>   candidate_links$sp <- as.numeric(as.vector(sp[,2]))
```

3.6 Output

The output of the algorithm is also a SpatialPointsDataFrame containing the matched coordinates of each GPS position and the corresponding data. In addition, for each position the OSM ID of the link it is matched to is included.

4 Testing

This section talks about testing of the above implemented fuzzy logic map matching algorithm. It describes the data used and the results of the testing.

4.1 Data

As already mentioned the test data comes from the enviroCar project. It is real field data submitted by users of the project. Unfortunately not all user submitted trajectories can be used for testing, because not all of them include the necessary data (HDOP, GPS.Bearing and GPS.Speed) required by the algorithm. The reason for this is that the recording of this data was not implemented in former versions of the app. So everyone using an outdated version off the app will continue uploading trajectories without this data.

An additional problem is that sometimes there are missing values in the data. This mainly occurs for the bearing if the speed of the vehicle is very low or zero and leads to an error in the FIS. The problem can be solved by just replacing the missing values with zeros, which of course can lead to incorrect matchings.

A positive aspect is, that there is data available for both, rural and urban areas, including complex structures such as roundabouts, complex urban road networks and motorway junctions, which is very suitable for testing. The trajectories used for testing all lie in or around the city of Münster, because this area contains almost all available trajectories.

The map data used for testing is acquired from OSM as described above.

4.2 Results

Since only the user submitted trajectories are available without any corresponding true data for the vehicle locations it is of course hard to tell if the algorithm finds the correct links. So what we can do is looking for obvious mismatches and in what situations they occur.

In rural areas the map matching algorithm works very well (Figure 6). The road density is low and the GPS positions are often more accurate than in urban areas, which makes the identification of the correct links easier.

Of course the vehicle position on the link is often not very accurate because the nearest point on the link is chosen as the vehicle's location on the link.

Link identification in urban areas also works surprisingly well. Even though the road density is much higher than in rural areas and the GPS positions often lie way off the the actual roads, the algorithm still manages to identify the correct links as shown in Figure 7. But here you can also see a problem associated with the way the vehicle's position on the link is determined. Even if the algorithm may identify the correct link, the position on the link often becomes the end node of the link, if the GPS position lies ahead of the link.

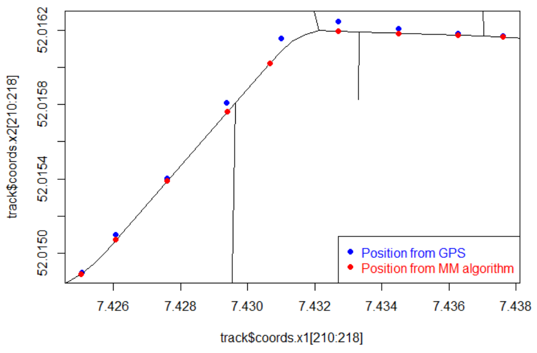

Figure 6: Map matching in rural areas

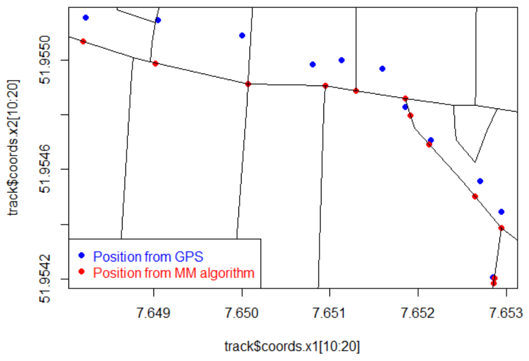

Figure 7: Map matching in urban areas

In general most of the identified links look correct or proper. Even more complex structures such as roundabouts are normally no problem. Obvious mismatches only occur in very complex structures with lots of roads and bends on a small area such as motorway junctions, which can be seen in Figure 8. Sometimes they also occur on roads with several lanes, where some positions get matched to the wrong lane and of course if values in the data are missing.

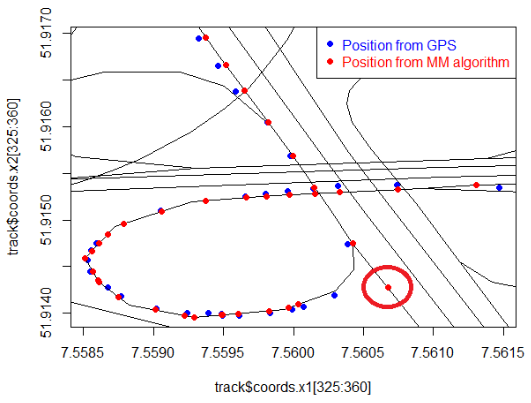

Figure 8: Mismatch on a motorway junction

5 Conclusions and Recommendations

The main goal of this thesis was the implementation of reliable map matching algorithm to support the enviroCar project. This Chapter summarizes the results obtained in this research. Additionally some recommendations are made regarding the problems with the GPS trajectories obtained from enviroCar and regarding improvements of the here implemented algorithm.

5.1 Conclusions

In this thesis a fuzzy logic map matching algorithm, suitable to take care of the map matching in the enviroCar project, was implemented. This algorithm was chosen because of its very good performance in terms of link identification. It was successfully implemented in R according to its description in Quddus (2006). However, some small changes were necessary.

As described in Chapter 2 and 3 it was not possible to calculate the VCV associated with the errors in the computed GPS position because the data to do so was not obtainable from the GPS trajectories. This caused modifications in some parts of the algorithm such as the creation of the error region and the determination of the vehicle's position on a link. This means the error region, used for identification of candidate links, was implemented with a fixed size and the vehicle's position on link is simply determined using the nearest point on the link. Of course this is not an optimal solution but it does not influence the algorithms primary function, the identification of the links the vehicle is traveling on.

So the result of this thesis is an `R` package that is capable of matching GPS trajectories to an OSM road network using a fuzzy logic based map matching algorithm.

Finally, it is important to note that although the here implemented algorithm is working there are still some issues to look into. Problems at the moment are that the algorithm is not usable with all available trajectories of enviroCar, because some of the trajectories do not include all required data and another problem is that the acquisition of the road data can be very time consuming.

5.2 Recommendations

As already mentioned the main motivation for the implementation of the map matching algorithm was the enviroCar project. At the moment not all available trajectories include all the necessary data for the algorithm to work because of trajectories uploaded from former versions of the enviroCar app. These trajectories could be removed or flagged to solve this minor problem.

In future app versions data to obtain the VCV, which can be used to derive the error rectangle and to better estimate the vehicle position on the link could be included into the trajectories. Of course this is something the enviroCar developers have to decide.

An improvement to the performance of the algorithm could be a different source to obtain the road data. As described in Chapter 3, at the moment, the algorithm needs to download and process the data each time it is executed to create a digital road network. This can be very time consuming depending on the size of the area and the amount of features included in the area. If the working area is known, a digital road network prepared in advance could greatly increase the performance in terms of run-time.

References

Bivand, Roger, & Rundel, Colin. 2013. *rgeos: Interface to Geometry Engine - Open Source (GEOS)*. R package version 0.3-2.

Bivand, Roger S., Pebesma, Edzer, & Gomez-Rubio, Virgilio. 2013. *Applied spatial data analysis with R, Second edition*. Springer, NY.

Csardi, Gabor, & Nepusz, Tamas. 2006. The igraph software package for complex network research. *InterJournal*, **Complex Systems**, 1695.

DoD, U.S. 2008 (September). *GLOBAL POSITIONING SYSTEM STANDARD POSITIONING SERVICE PERFORMANCE STANDARD*. Tech. rept. 4. Assistant secretary of defense for command, control, communications, and intelligence.

Eugster, Manuel J. A., & Schlesinger, Thomas. 2010. osmar: OpenStreetMap and R. *R Journal*. Accepted for publication on 2012-08-14.

Hijmans, Robert J. 2014. *geosphere: Spherical Trigonometry*. R package version 1.3-8.

MathWorks. 2013a. *Foundations of Fuzzy Logic*. `http://www.mathworks.de/de/help/fuzzy/foundations-of-fuzzy-logic.html#bq3hn3k`. Web. 28 Feb. 2014.

MathWorks. 2013b. *What Is Sugeno-Type Fuzzy Inference?* `http://www.mathworks.de/de/help/fuzzy/what-is-sugeno-type-fuzzy-inference.html`. Web. 28 Feb. 2014.

Quddus, Mohammed A. 2006 (January). *High Integrity Map Matching Algorithms for Advanced Transport Telematics Applications*. Ph.D. thesis, Imperial College London, United Kingdom.

Quddus, Mohammed A., Ochieng, Washington Y., Zhao, L., & Noland, Robert B. 2003. A general map matching algorithm for transport telematics applications. *GPS Solutions,*, **7**(3), pp. 157–167.

R Core Team. 2013. *R: A Language and Environment for Statistical Computing*. R Foundation for Statistical Computing, Vienna, Austria.

Riza, Lala Septem, Bergmeir, Christoph, Herrera, Francisco, & Benitez, Jose Manuel. 2013. *frbs: Fuzzy Rule-based Systems for Classification and Regression Tasks*. `http://CRAN.R-project.org/package=frbs`.

White, Christopher E., Bernstein, David, & Kornhauser, Alain L. 2000. Some map matching algorithms for personal navigation assistants. *Transportation Research Part C: Emerging Technologies*, **8**, 91–108.

Acronyms

ATTS	Advanced Transport Telematics Systems
FIS	Fuzzy Inference System(s)
GPS	Global Positioning System
HDOP	Horizontal Dilution of Precision
HE	Heading Error
IMP	Initial Map Matching Process
MF	Membership Function(s)
OSM	OpenStreetMap
PD	Perpendicular Distance
SMP	Subsequent Map Matching process
SMP-1	Subsequent Map Matching process along a link
SMP-2	Subsequent Map Matching process at a junction
VCV	Variance-covariance matrix