Dynamic Routing Using Precipitation Data

Philipp Kisters, Dirk Bade and Julius Wulk
University of Hamburg,
Germany
Email: {kisters,bade,1wulk}@informatik.uni-hamburg.de

***Abstract*—Smart mobility is one of the cornerstones of smart cities and in particular, smarter routing decisions are desired and required in order to optimize urban, intermodal transports for individuals as well as businesses. Smart routing comes as a multi-criterion optimization problem, not only dealing solely with efficiency anymore but also with ecologic and economic aspects, as well as comfort, safety, fun etc. In this paper, we propose two new routing algorithms that dynamically re-evaluate rainless routes using up-to-date, high-resolution precipitation information, that additionally take user preferences into account and also consider optimal temporal offsets for departure. Thereby, we not only focus on, e.g., cyclist and pedestrians, who would obviously benefit from such routing, but also on more visionary use cases including aerial or water-borne vehicles. We implemented a middle-tier and a mobile application in order to proof the feasibility of our precipitation-based routing algorithms and demonstrate the advantages by presenting preliminary results of an evaluation using real historical precipitation data.**

***Index Terms*—Vehicle routing, Dynamic Routing, Intelligent Transportation Systems, Smart City**

I. Introduction

In recent years, traffic congestions and the resulting ecologic and economic impacts became a major problem for metropolitan areas [1]. Local governments face these problems by promoting alternative transportation modalities, e.g. public transport, bicycles, e-scooters and even more visionary: self-flying taxis or delivery drones. However, one important factor for still preferring the car over such alternatives are the current weather conditions. There is a direct relationship between bad weather and increased usage of cars [2]. But oftentimes, rain or snowfall is a local phenomenon and only affects a small demarcated area. We claim that if it would be possible to circumvent these areas along a route from a starting point to a dedicated destination, citizens will be more motivated to swap modalities, leaving the car in the parking lot (cf. [3]). Also aerial or water-borne vehicles, vehicles with open loading areas, cabriolets, pedestrians as well as any other transport user might arrive at their destination even more safely and securely when avoiding bad weather conditions.

Hence we propose new routing algorithms that take precipitation data into account in order to find a rainless route either spatially (alternative route) or temporally (alternative departure time) while taking the individual preferences of users (e.g. favor short route over rainless route) into account. One routing algorithm serves aerial and water-borne vehicles, which are not restricted to road courses and follow a more bee-line route (*area routing*), the other algorithm outputs classical pathways mapped onto streets (*street routing*). Challenging for both algorithms are the dynamics of weather conditions and the need to possibly adapt the routes as time progresses while still adhering to the users' preferences.

We demonstrate the feasibility of the proposed algorithms by presenting our prototypical implementation of a middle-tier and a mobile app that calculate rainless routes based on precipitation data of an X-band radar with a radius of 20km, a spatial resolution of 60m and a temporal resolution of 30s in the City of Hamburg, Germany [4] and evaluate these using historical precipitation data.

The remainder of this paper is organized as follows: Section II provides a short overview about related approaches. In Section III requirements for the routing system (algorithms, middle-tier, mobile app) are identified, followed by the conceptual foundations of both routing algorithms in Section IV. Afterwards, Section V presents a bird's eye view on the architecture of the routing system, providing more detailed insights into our prototypical implementation, which we subsequently analyze in Section VI. Finally, Section VII discusses our approach and Section VIII summarizes the paper and gives prospects for future work.

II. Related Work

Several works exist that incorporate additional (dynamic) data into routing recommendations. Here, we have to distinguish between static environmental data and dynamic data. Static data is used, e.g., by Franke et al. [5] who consider inclination of roads in order to generate efficient routes for battery-powered electronic vehicles (e.g. wheelchairs). In [6] the authors additionally make use of data about obstacles like stairs and curbsides to help disabled people find accessible routes. The GIScience Research Group from the University Heidelberg published a blog post [7] describing a 'green' routing approach that guides users through recreational areas. Similar to this, the *Smart GraphHopper* project [8] aims at finding routes based on a non-realtime participatory sensing approach considering stressors like noise and air pollution.

Dynamic data is (additionally) used, e.g., in the *SecureRouting* project [9], in which data about road accidents are combined with current weather data to warn users following the same route under similar weather conditions. Likewise, in [10] risks are avoided by incorporating current traffic conditions as well as user preferences into a cost function used to generate appropriate routes. The *RouDy* project [11] also uses precipitation data in order to circumnavigate passages in forested areas that suffer from drowning or fallen trees.

978-1-7281-1796-6/19/$31.00 ©2019 IEEE

Even more sophisticated is the routing approach in [12] which not only optimizes the route but also suggests optimal vehicle operation strategies depending on current time, remaining fuel, driving behavior as well as traffic and weather conditions.

All of these routing approaches have in common, that not only the distance between two points is used to find an optimal route, but additional (dynamic) information is considered in the cost function such that the resulting routes might not be optimal in regard to distance, but better suit the needs of users and envisaged use cases. In our approach, we target even more use cases offering additional bee-line area routing as an alternative to street routing. We use more fine-grained dynamic geographic data with a very high spatial and temporal resolution as compared to any other related work. Furthermore, we respect user preferences when mapping the weights in routing graphs and adapt these accordingly. And we consider the option to delay the departure if waiting for less precipitation turns out to be beneficial.

III. Requirements

We conducted a requirements analysis among potential developers as well as users and identified requirements in three different categories: i) mobile app, ii) middle-tier, and iii) route calculation.

Within the mobile app, it should be possible to define routing preferences like a max. temporal offset for departure, the max. tolerable precipitation and the max. detour one is willing to make. For this, we require an easily understandable representation of precipitation values and patterns, because users might not be familiar with the respective units. Moreover, the app should use up-to-date precipitation data and not make extensive use of the mobile device's resources like CPU, memory, energy, and communication.

The middle-tier, responsible for gathering precipitation data from different data sources, is expected to be resource-rich. Therefore, it should request precipitation data from the sources as often as fresh data is available. Due to possible heterogeneous representation formats, the middle-tier should transform all source data into an internal data structure to allow a uniform and efficient access later on.

The route calculation should, of course, yield optimal results with respect to distance, amount of precipitation and user preferences. The calculation should predict user movement and hence use nowcasting data (precipitation forecasting for short time frames) as well as interpolations between data points and take place repeatedly in order to take current weather trends for the predicted area into account.

For evaluation purposes as well as for specific use case adaptations, the routing algorithms should be configurable (e.g. adjust weights on nodes and edges) and easily replaceable (e.g. *A** vs. *Dijkstra*). Finally, an appropriate metric should be developed in order to compare routes.

IV. Rainless Routing

As stated earlier, users might want to choose between different kinds of routing depending on the use case: area routing or street routing. Both approaches work on a two-dimensional array of tiles. Each tile represents an area demarcated by two geo-positions and holds a list of current and forecasted precipitation values. In the following, we will discuss the respective algorithms and corresponding challenges.

A. Area Routing

The area routing is designed for non-street-bound use cases, e.g. for aerial and water-borne vehicles or in general for open spaces without any streets.

On an incoming routing request, all relevant tiles are loaded from a data storage. Tiles are relevant if they are located between the start and the endpoint. Additionally, a buffer area is considered, whose extent depends on the user's preference for the maximum detour. The relevant tiles are used to calculate the optimal route using an arbitrary search algorithm like Dijkstra or A*. Each tile represents a node and the transition from one node to a neighboring node is associated with a cost corresponding with the neighboring node's precipitation value (see Figure 1).

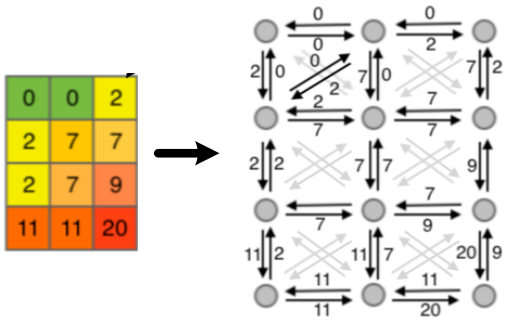


Figure 1. Mapping Precipitation Values to a Graph

An additional weight is imposed on an edge depending on the user's preference for detours. In this case, edges leading away from the envisaged destination node might have a higher weight resulting in a higher cost. If a user is not willing to make any detour, the weights for nodes leading away from the destination are weighted so high that a more bee-line route will be generated ignoring lower precipitation in neighboring tiles on the way.

B. Street Routing

For short ways or areas that are well known by the user, areal suggestions might help even if one is bound to the road network. But for other users following the optimal bee-line for a route, e.g., through an unknown city, might be quite annoying, because it is pretty much like using a compass when required to follow a street. Moreover, in some scenarios, the bee-line is just not applicable due to large barriers that cannot be crossed (e.g. lakes, rivers or large buildings). For such cases, we also developed a street routing algorithm that projects the precipitation onto a street graph and then uses a search algorithm to find the optimal route.

For such a projection, two fundamental options are available: The precipitation data contained in our tile maps can

either be used as weights on nodes (geolocations connected by streets) or on edges (streets). Weighting the nodes is more efficient due to a smaller amount of nodes in a street graph, but during route calculation, only the values of the tile in which the node is located in would be considered. The case that a street might traverse multiple tiles with varying precipitation values is not taken into account and thus results in a less realistic representation. On the other hand, weighting the edges will need more processing, simply because there are much more edges in a street graph and more special cases need to be taken into account. Considering such special cases, two important questions arise:

- Which value is assigned to an edge if multiple tiles are traversed?
- Which value is assigned when following "long edges"?

If an edge is completely located within one tile it can easily inherit its precipitation value. If it connects two tiles, the weight can be set to the value of the tile that will be entered (like in the area routing). But oftentimes edges span multiple tiles. Hence, to answer the first question, we experimented with two different methods for assigning a weight to the edge: We used the *Nearest Neighbor* algorithm (NN) for a simple and the *Inverse Distance Weighting* algorithm (IDW) for a more realistic representation. NN chooses the tile that contains one of the main parts of the edge, accepting that this tile has a completely different value than the other tiles that are traversed by the edge. Using IDW all tiles that are traversed are taken into account according to their share of the edge, which yields a more realistic weighting.

Interpolation is also needed when the graph search reaches a node and the corresponding expected user's arrival time at this node lies in between the validity range of two successive precipitation values (in our prototype the temporal resolution of tile maps is currently fixed to 30 secs). Here as well, the closest value (NN) or an interpolation with IDW using the two surrounding values can be used.

Another case to consider are "long edges". Think of an edge representing a long street without any crossings. Driving on this street probably takes more time than the temporal resolution of tile maps. Which weight shall be assigned to this long edge? One approach (depicted in Figure 2) would be to internally split the edge into subedges and sum up the precipitation value for each subedge depending on the expected arrival time of the user and the corresponding future precipitation value in the appropriate tile. The sum of precipitation values is then taken as a weight for the complete edge. While it solves the challenge in the most realistic way, it adds additional complexity to the routing algorithm. To avoid this we again use the inverse distance weighting algorithm to gain an interpolated value over the neighboring tiles, which is finally multiplied by the length of the edge to represent the edge's overall precipitation.

Finally, once the street graph has been weighted, classical algorithms for finding a shortest path like Dijkstra or A* are used to get an optimal route.

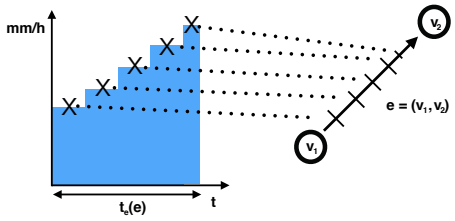


Figure 2. Summing up Precipitation on Long Edges

C. Delayed Departure

Are the current precipitation conditions so bad that the user is not satisfied with the resulting route, the departure may as well be delayed. This approach can be used with both routing algorithms. Instead of starting the route immediately, a temporal offset is used to check if upcoming precipitation conditions allow for a rainless route. For this purpose, the user may set a maximum tolerable departure delay, which is then used to run multiple routing calculations and find the optimal departure time.

V. Architecture

We implemented the presented routing algorithms in order to prove the feasibility of our approach. The general architecture is depicted in Figure 3. On the left-hand side we have multiple possible inputs, e.g. the X-band radar for productive usage, a server with historic precipitation data for evaluation purposes or a distributed sensing network used in our decentralized approach (cf. Section V-B). The import of precipitation data takes place continuously, depending on the update interval. In the case of the X-band radar, for example, the data is pulled every 25 seconds from the radar's server.

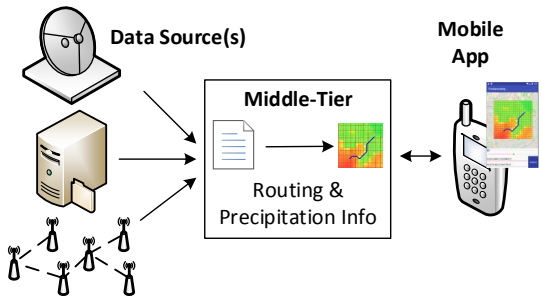


Figure 3. General Architecture of the Rainless Routing Approach

On the right-hand side, we have a mobile app, which shows a map and an overlay with current precipitation data surrounding the user's location. Users may set preferences according to their needs (e.g. prefer rainless routes to short routes) and finally set start and end locations for the route calculation. The route calculation takes place in the middle-tier. While on the way, the user continuously requests route updates by calling the service. Update intervals may be intelligently adapted by the mobile app in regard to the nowcasted rain probability or the user's need to save mobile data. The interface of the

middle-tier mainly offers access to two resources: i) the precipitation info and ii) the calculated route info. The access may be parameterized in order to demarcate the area of interest, but also to provide user preferences to route calculation, to choose the type of routing (area or street) and the used routing algorithm (A*, Dijkstra) - cf. Section III.

For the middle-tier, we experiment with two completely different approaches. In the following, we first detail a centralized middle-tier solution. As a result from the experiences we made, we afterward also briefly sketch a completely decentralized approach with a multitude of distributed data sources as well as processing services, which are choreographed in order to account for scalability affordances.

A. Centralized Architecture

The middle-tier is logically centralized (though it may be physically distributed) and consists of five main building blocks as depicted in Figure 4:

- To account for multiple heterogeneous inputs, the imported precipitation data is pre-processed by the *Data Preparation* module and transformed into an internal data model used for uniform and efficient retrieval later on.
- A lightweight (in-memory) *Data Store* is used for efficient access of data objects.
- *Precipitation Info* simply retrieves precipitation data from the data store to serve the routing services and the mobile app (for visualization).
- The *Routing Algorithms* in turn calculate rainless routes based upon the precipitation data. The street-based routing moreover makes use of a map in order to generate a street graph consisting of nodes and edges upon which the algorithm acts.
- Results are post-processed and transformed into an efficient data format before being sent back to the mobile app.

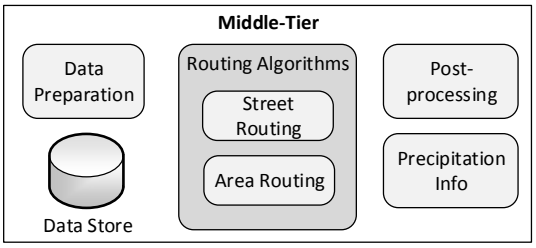


Figure 4. Centralized Architecture of the Middle-Tier

B. Decentralized Architecture

Going one step further, we envision a completely decentralized scenario with a multitude of heterogeneous sensors, most of them privately operated, e.g. in cars or smart gardens, delivering streams of precipitation data to processing and storage services run as part of a citizen-driven distributed smart city data space, namely *SANE* (Smart Network for Urban Citizen Participation [13]). The SANE project offers appropriate abstractions and methods to easily share (sensor-) data between parties, thereby respecting privacy and ownership and allows for processing and storage of this data within dynamic peer-to-peer networks.

In this context, we have *SANE Sensors* that represent real or virtual sensor devices delivering streams of precipitation data. And we have *SANE Nodes* that offer storage and processing services within the network. The building blocks of the centralized approach presented in the previous section are now distributed within the *SANE Network* as individual services running on one or more SANE nodes (see Figure 5).

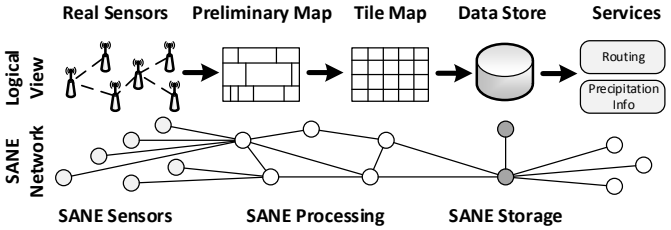


Figure 5. Decentralized Architecture of Rainless Routing

Due to the distributed nature of SANE sensors, we need to consider additional pre-processing steps in order to generate a tile map with precipitation data (cf. Section IV). Challenging are the heterogeneity of these sensors because they do not provide concrete precipitation data in millimeter units, but something arbitrary, which has to be abstracted to precipitation classes (e.g. no, low, moderate, high, very high precipitation).

At first, such data has to be collected from the sensors and a preliminary tile map has to be generated. As not every tile will be backed by a sensor and sensors are probably not equidistant the resulting map needs additional interpolations and aggregations to homogenize the tiles and the precipitation values. Moreover, a service to create a nowcast out of a series of (past) tile maps is required. In the centralized scenario, this task has been done by the radar's backend and now needs to be provided within the network.

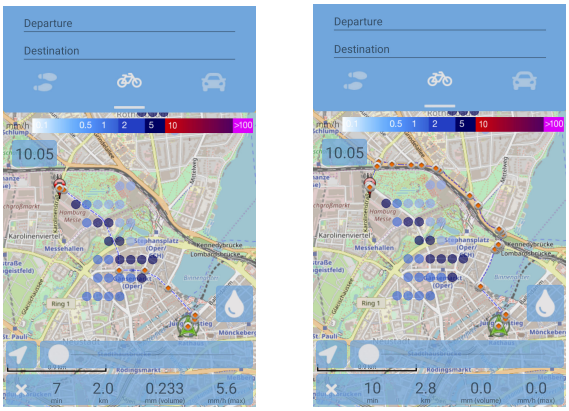
Since the pre-processing services are not affected by the number of requests, they do not need to scale. Multiple instances for each of the pre-processing steps are only required for redundancy or if the number of SANE sensors exceeds a certain threshold. In the latter case, a divide & conquer mechanism needs to be established that allows for task splitting (by division of subarea responsibilities) and result merging.

The final tile map corresponds to the output of the data sources as described in the centralized architecture (cf. Section V-A) and will be stored in SANE storage nodes. Once a new routing request arrives in the SANE network, it is forwarded to an instance of an appropriate routing service, which subsequently retrieves the stored tile map and starts route calculation. As routing services are stateless and all required parameters are passed by the clients, the services can easily be replicated within the network. A further distribution of routing subtasks is possible, but currently not planned.

C. Implementation

All components of the architecture have been prototypically implemented (despite the decentralized architecture, which we are currently working on). The import of precipitation data is done by pulling an FTP server every 25 seconds in order to get a *NetCDF*[1] file containing all required information. The services in the middle-tier are realized as microservices running in a Java VM offering a REST interface for the client. To calculate rainless routes, we extended the *Graphhopper* routing engine [14] to use precipitation data as additional weight. For the generation of the street graph, the implementation relies on the *Open Street Map* (OSM) project.

The client is realized for the Android platform. To display calculated routes we are using the *OSM-Droid* library [15]. Displaying precipitation values intuitively is challenging and giving the user an obvious indication of the rain en route we chose a layered approach in which colored overlays are projected onto the OSM-Droid map tiles. The source code of our implementation is publicly available as open source[2].



(a) Using Shortest Path (b) Using Rainless Path

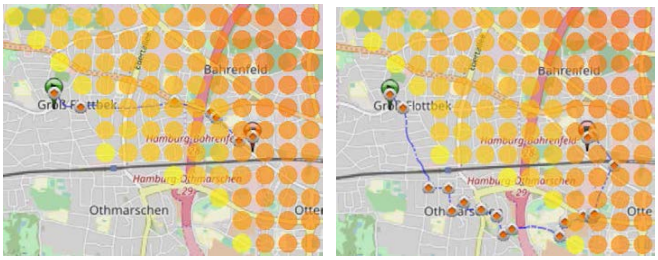
Figure 6. Screenshot of Android Application

VI. Analysis

We conducted a preliminary analysis in order to underline the advantages of our rainless routing approach. The analysis is based on a recording of historical precipitation data in order to ensure repeatability. At first, a qualitative analysis based on a visual comparison between classical and rainless routing is presented. Afterward, we briefly sketch some findings of a small quantitative analysis. An extensive evaluation is subject for future work.

A. Qualitative Analysis

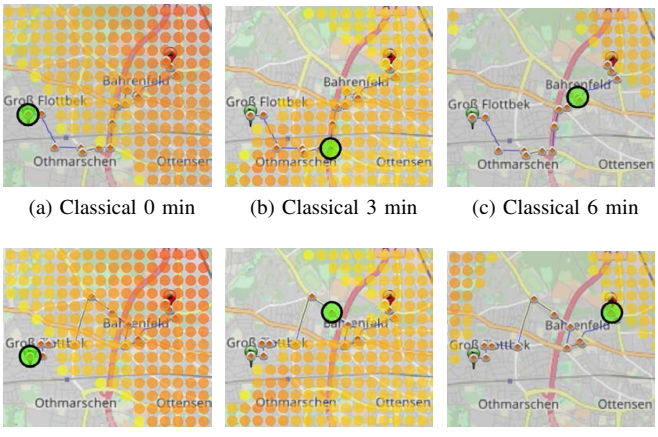
The qualitative analysis is simply based on a visual comparison of resulting routes. In Figure 7 we compare a classical routing solely based on the shortest path with our rainless approach. To ease interpretation we completely omitted the time aspect, i.e. the vehicle movement as well as precipitation evolution. It can be seen that while classical routing directly leads on the shortest path (represented by a thin blue line) through the rainy area (represented by colored circles), the rainless routing yields a path that gets us immediately out of the rainy area and then a way round towards the destination.



(a) Classical Routing (b) Rainless Routing

Figure 7. Comparison of Classical and Rainless Routing Omitting Time

In a realistic setting, one would, of course, want to consider temporal aspects, in particular, the movement of one's own vehicle as well as the evolving precipitation pattern. In Figure 8 we again contrast the classical routing with our rainless routing approach, but this time we use nowcasting data to foresee the precipitation pattern in a couple of minutes. The classical routing starts on the left (big green circle) and leads on the shortest way downwards, where it is pouring in minute 3. The rainless routing foresees this incident and chooses an upwards route, which is slightly longer but circumvents the expected rainy area.



(a) Classical 0 min (b) Classical 3 min (c) Classical 6 min

(d) Rainless 0 min (e) Rainless 3 min (f) Rainless 6 min

Figure 8. Comparison of Classical and Rainless Routing omitting Time

B. Quantitative Analysis

For the preliminary quantitative analysis, we picked 15 different starting points and destinations and chose pairs with an approximate distance of 3km, 5km, and 10 km. Using the same precipitation data sets as for the qualitative analysis, we conducted several experiments using different routing algorithms (Dijkstra vs. A*), different temporal and spatial interpolation methods (nearest neighbor vs. inverse distance

[1] *Network Common Data Format*, used to exchange scientific data.
[2] https://github.com/rainlessrouting

weights with different radiuses), different delta-values for assessing measuring points and different time-windows for departure and logged, i.a., the average and maximum amount of precipitation per route.

For the sake of brevity, only one experiment is depicted in Figure 9. Here, the average amount of precipitation (in mm) is presented in regard to different radius used by the inverse distance weight function. The routing algorithm was based on Dijkstra and used weights on edges. The result clearly shows that rainless routing without considering time (upper green circles) does not perform much better than classical routing (green diamonds, *NN-Dijkstra* and *NN-ALT*), but considering time is beneficial for the rainless routing (lower blue circles). The blue triangle at the bottom right corner represents the average amount of precipitation when using an optimal departure time within a 10 min window (*IDW m.Z.*). When increasing the radius of the inverse distance weighting, the results stabilize, but at the costs of higher execution times (not depicted here).

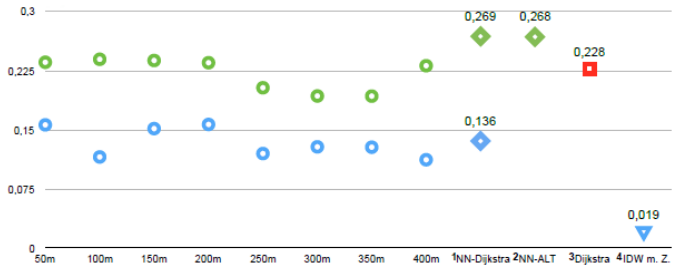


Figure 9. Comparison of Routing Strategies

VII. Discussion

As detailed in the previous section, we conducted an analysis of our application with historical data. Since the used X-band radar infrastructure currently does not feature nowcasts we do not have any outdoor experience, yet. Although the preliminary results are promising, evidence of the practical effectiveness is hence lacking.

In addition, efficiency in terms of execution times needs to be further optimized. Currently, the projection of precipitation data is done with every single request. Once we aim for productive usage, at least a simple caching strategy is required to increase user experience. In the long run, the presented decentralized approach would also tackle this problem.

VIII. Conclusion

There are plenty of good reasons to include precipitation data into routing decisions: comfort, safety, and sustainability. And it is for the good of all traffic participants, be it pedestrians, cyclists, motorists etc. There already are works that integrate weather data into routing decisions, but in this paper, we presented two approaches that work with high-resolution precipitation data and in combination cover a wider range of use cases. Moreover, we sketched a centralized as well as a visionary decentralized architecture for route calculations. The former has already been implemented and served as a prototype for a first analysis that already indicates that our rainless routing is indeed beneficial.

But we still have some steps to go: Among our prospects for future work is a link-up to live data from the X-band radar to sample our mobile application in practice. On the other side, we are looking forward to using the extensive amount of high-resolution historical precipitation data to conduct a comprehensive quantitative evaluation in order to back our impressions with more empirical data. Here it would also need to be taken into account how the usage of these routes will effect the traffic. Could the amount of cyclist in specific roads increase in a way that it would effect other traffic participants? Additionally, we already started working on the implementation of our decentralized architecture and as we already logically separated pre-processing and routing in our decentralized vision, we are eager to also integrate other kinds of dynamic (real-time) data, e.g. noise or air pollution.

IX. Acknowledgement

We would like to thank our colleagues, in particular Akio Hansen, from the Meteorological Institute in the University of Hamburg for inspiration, discussion and the access to the X-band radar data.

References

[1] I. Research, "Global traffic scorecard," INRIX, Tech. Rep., 2018.
[2] M. Sabir, "Impact of weather on daily travel demand," VU University, Department of Spatial Economics, Tech. Rep., 2010.
[3] M. Klafft, "Including weather forecasts in routing decisions of navigation systems for road vehicles: The users' view," in *36th International Conference of the Chilean Computer Science Society, SCCC 2017, Arica, Chile, October 16-20, 2017*, 2017, pp. 1–5.
[4] M. I. U. of Hamburg, "X-band radar," https://wetterradar.uni-hamburg.de/index.php?id=4035, 2019.
[5] D. Franke, D. Dzafic, C. Weise, and S. Kowalewski, "Konzept eines mobilen osm-navigationssystems für elektrofahrzeuge," *Angewandte Geoinformatik*, pp. 148–157, 2011.
[6] F. D. Džafić D., Schoonbrood P. and K. S., "enav: A suitable navigation system for the disabled," *Ambient Assisted Living. Advanced Technologies and Societal Change*, pp. 133–150, 2017.
[7] GIScience Research Group Heidelberg University. Introducing healthy routing preferring green areas with openrouteservice. [Online]. Available: https://www.directionsmag.com/pressrelease/4691
[8] V. Nallur, A. Elgammal, and S. Clarke, "Smart route planning using open data and participatory sensing," in *IFIP International Conference on Open Source Systems*. Springer, 2015, pp. 91–100.
[9] G. Kiechle, T. Markmiller, and M. Obermayer, "Securerouting-sicheres routing unter verwendung von unfallhäufungs-und wetterdaten," *Angewandte Geoinformatik 2013*, pp. 444–449, 2013.
[10] R. Hoyer, "Verkehrssicherheit als bestandteil einer dynamischen navigation in straßennetzen," in *19. Conference of Traffic and Transportation Sciences*, 2003.
[11] B. SCHMIDT, S. BÖNIGK, and M. STÖCKER, "Roudy: Ein ansatz zur situationsabhängigen routen-und navigationsplanung," 2013.
[12] N. Ogulenko, S. Frey, J. Nahm, and M. Rössle, "Cloud-basierte optimierung von fahrzeugbetriebsstrategien durch clustering mit genetischen algorithmen," *Informatik 2016*, 2016.
[13] H. Bornholdt, D. Jost, P. Kisters, M. Rottleuthner, D. Bade, W. Lamersdorf, T. C. Schmidt, and F. Mathias, "Sane: Smart networks for urban citizen participation," in *2019 26th International Conference on Telecommunications (ICT)*. IEEE, 2019, (accepted).
[14] G. GmbH. Graphhopper routing engine. (Last checked 2019-03-05). [Online]. Available: https://github.com/graphhopper/graphhopper/blob/master/README.md
[15] N. Boyd, M. Kurtz, A. O'Ree, and M. Kergall. (2018) Openstreetmap-tools for android. (Last checked 2019-03-05). [Online]. Available: https://github.com/osmdroid/osmdroid