15 Dec 2021

FINAL REPORT

PROJECT

Creation of a methodology and calculation solution for the modal distribution of the mobility of the city of Tartu

Project Details:

Name	Registration code
University of Tartu	74001073
Project Name Creation of a methodology and calculation solution for the modal distribution of the mobility of the city of Tartu	
Document Type	Final Report (Ver2.0)
Project Period (start - end)	19/05/2021 – 30/11/2021

1 INTRODUCTION

Modal-split is one of the complex tasks in transportation research. The main approach in the literature for extracting the modes of transport has been traditional surveys. Although splitting the different vehicle types, using data of city sensors has been studied widely, but the challenge arises when in addition to the vehicles, we want to understand the share of pedestrians and cyclists in daily urban mobility. The quality of the results in the latter case depends heavily on the details of the data collected by the sensors and available statistics as a base.

In collaboration with the City of Tartu, this project aims to extract the daily share of transportation modes in the city. We calculate the hourly district-level origin-destination matrices (OD-matrices) of each mode of transportation. The map of different districts of the city of Tartu is presented in the Fig. 1.

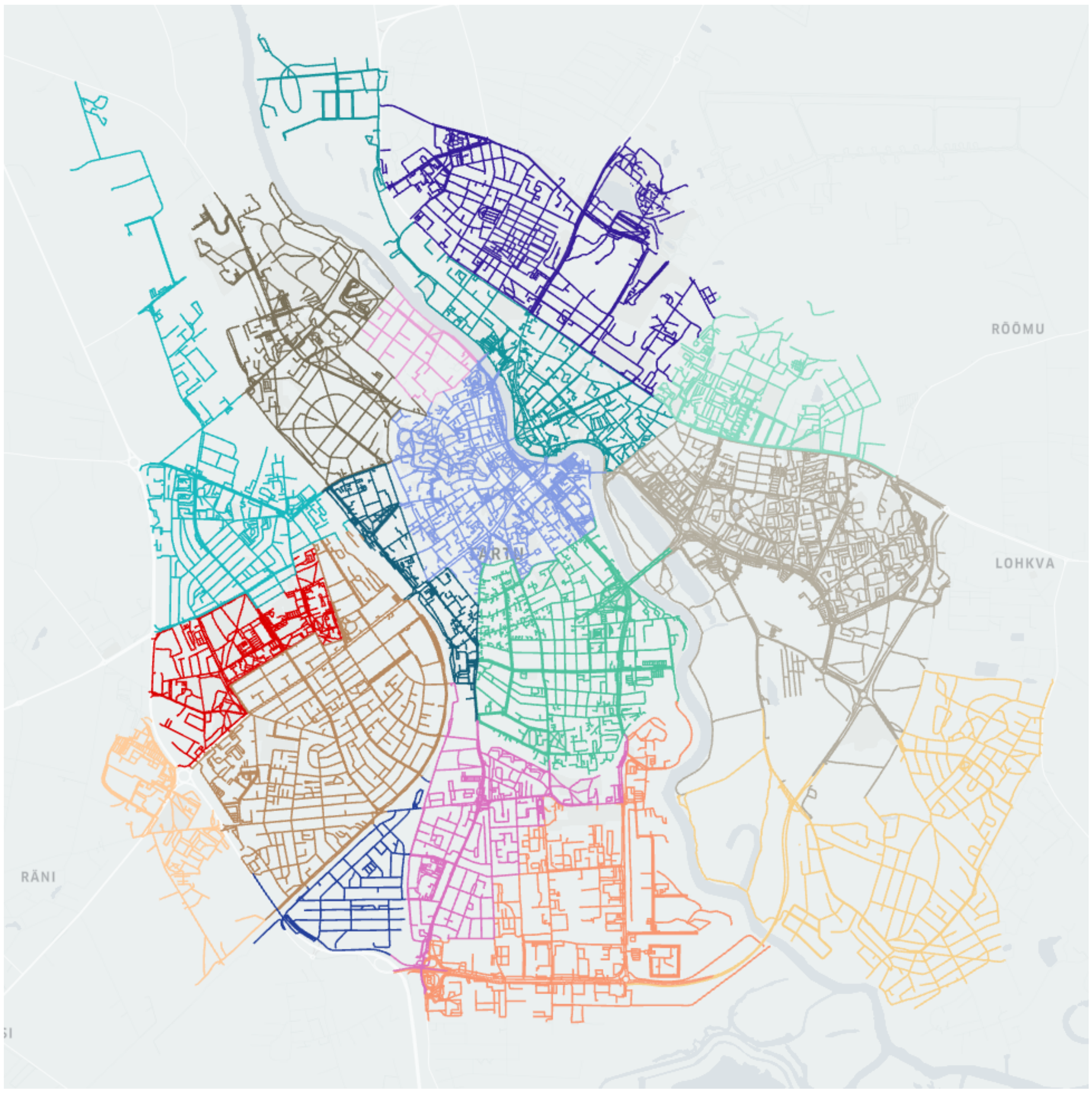


Figure 1. Different districts of city of Tartu

2 ARCHITECTURE

To perform the transport mode share estimation, we have developed a software solution comprising of two main blocks: the *data analysis pipeline* and *web dashboard*. The data analysis pipeline extracts fine-grained data (usually single-trip or single sensor-level granularity) from several modality-specific data sources (e.g., bus ticket validation infosystem). The raw data is analyzed to produce hourly and district-level estimates for the entire city while taking into account the directionality of the trips between districts. In other words, the output is an OD-matrix for each of the four transport modes for a one-hour period.

The web dashboard provides a graphical user interface in the form of a web application for visualizing and browsing the pipeline's results. Additionally, the dashboard provides additional aggregation functionality, such as presenting 24h period results based on the hourly pipeline outputs. The two blocks are linked by persistent storage: either a SQL database or simple disk storage. The *pipeline* is scheduled to execute once every 24 hours to analyze the previous day's information and store the results to make them available for the *dashboard*.

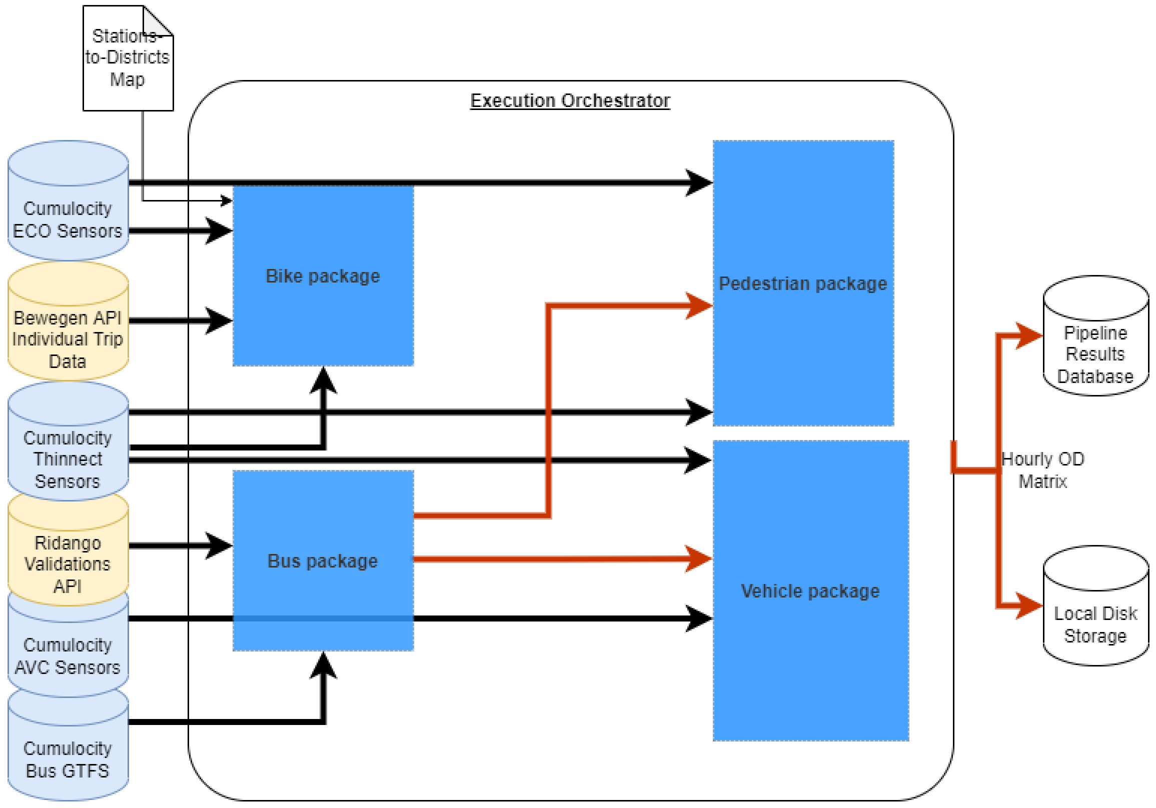


Figure 2. The architecture of modal split data analysis pipeline.

In the following sections we further detail the blocks and their integration. The high-level view of the architecture can be seen on Fig. 2.

2.1 Data analysis pipeline

The pipeline consists of four modules (packages); one per transport modality and an execution orchestrator. The orchestrator ensures the execution of individual modules in a specified order (necessary due to the interdependence between modules) and manages the re-execution of modules that produce failures.

2.1.1 Transport Modality Packages

Every modality-specific package follows the same basic design: it should fetch all necessary input data for its execution (either via HTTPS REST API calls or from local storage), perform the aggregation and analysis, and then save the results into persistent storage. If one package's input includes results of another package (e.g., as in the case of pedestrian package relying on bike package's output), the package assumes the execution orchestrator to have already finished execution of the necessary packages.

The specifics of each package are detailed in the rest of the report; a more detailed perspective of the interdependence of the packages and their internal logic can be seen in Fig. 18.

3 DATA SOURCES

Different data sources used as the packages' inputs are listed in Table 1. Data sources can be categorized into *dynamic* and *static*. Dynamic sources represent those whose data is fetched on-demand each time the pipeline is run. Static files are fixed instead of on-demand loading. Static files are used for data that are expected to change infrequently, such as the geo-information about Tartu district borders.

3.1 Dynamic sources

Dynamic sources expose their data to the pipeline through some HTTPS RESTful API. In the case of Tartu City's Cumulocity platform instance, three different types of sensor counts (ECO, Thinnect, AVC) and one kind of coordinate information (Bus GTFS) are used. ECO sensors count pedestrians and bicycles. Thinnect sensors count vehicles and light traffic[1]. AVC sensors count vehicles. Most of the sensors also consider the directionality of the movement (they count events for a certain direction of movement). Ridango API provides anonymous data about bus ticket validations. The validation events include information about the bus stop, the bus line, and the time of the event. Bewegen API provides data about individual bike trips using the city bike-sharing system. This includes the time and bike charging station where the trip was started end where it was ended.

The Cumulocity, Bewegen, and Ridango APIs provide the functionality to query data for a specific date.

3.2 Static sources

Static sources are the following files.

- A file depicting the geographic shapes/borders of the districts, used to identify which trip originated/destinated in which district. The file also assigns an identifier code to each district
- A mapping from City Bike charging stations to districts.

4 METHODOLOGY

In this section, the detailed methodology used in each package is discussed. Every package (Fig. 2) outputs an hourly district-level OD-matrix which corresponds to one mode of transportation.

[1]light traffic: both pedestrians and bicycles, undistinguished.

Data Source	Packages	Type
Cumulocity ECO Sensor Measurements	Bike, Pedestrian	Dynamic
Cumulocity Thinnect Sensor Measurements	Pedestrian, Vehicle	Dynamic
Cumulocity AVC Sensor Events	Vehicle	Dynamic
Cumulocity Bus GTFS Events	Vehicle	Dynamic
Bewegen REST API	Bike	Dynamic
Ridango REST API	Bus	Dynamic
peatus.ee Bus Stop Information	Bus	Dynamic
Bicycle Charging Station to District Map	Bike	Static File
Tartu Districts Geodata File	All packages	Static File

Table 1. Data sources of Modal Split pipeline. "Cumulocity" refers to Tartu City Cumulocity

4.1 Bus Package

The goal of the bus package is to estimate the district-level OD-matrix. All bus users in Tartu are expected to scan their tickets at the entrance to the bus. Therefore, the system stores the origin stop for each passenger automatically. However, the challenge is estimating the trip destination since there is no *e*xit record stored for the bus passengers. Fig. 3 presents a more detailed architecture of the bus package. The output of the bus package is a district-level, dynamic OD-matrix that will be used for initialization in vehicle and pedestrian packages.

4.1.1 Algorithm

In this part, different steps of building a dynamic OD-matrix for bus passengers, are explained.

Data acquisition. The bus package uses two sources for retrieving the data. *Ridango Api* makes the majority of our input data. It provides daily information about public buses and their routes, trips, and passenger validations on each bus. Information about stops is extracted on a daily basis from *peatus.ee*. As the first step of the algorithm, all of this data is downloaded through HTTP requests.

Choosing potential stop station for each passenger. Given the trip origin, the potential stop stations for each passenger is listed. This list contains all the following bus stations in the same bus route, excluding the stations that the bus has not stopped. *Ridango Api* provides the information on whether or not a bus has stopped at each stop of its route. In the end, we choose the stop station for each passenger uniformly at random from the list of potential stop stations for that passenger.

Building OD-matrix. After having the start and the stop station for each bus passenger, the stations are mapped to the city districts, and the district level hourly and daily OD-matrices are constructed. These matrices present the mobility flow of bus users in the city and will be used later as input for pedestrian and vehicle packages.

4.1.2 Lab Validation

Bus input data directly retrieve information from scanning devices located in buses. Therefore, the number of ticket validations per day reflects the number of bus trips with very high accuracy. Hence validating the number of trips is not relevant in this package.

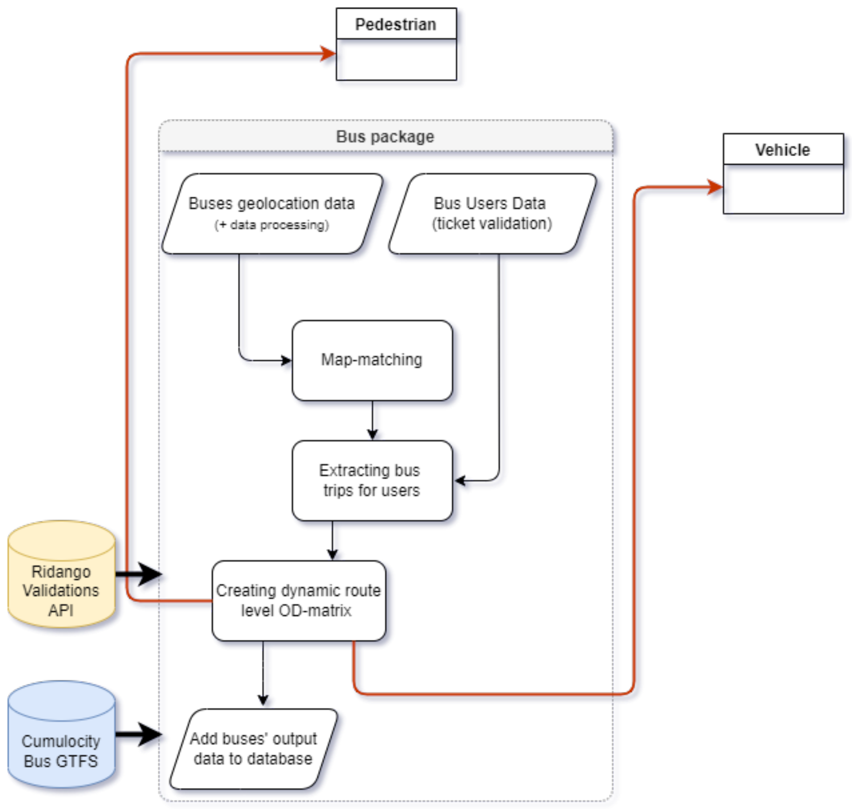


Figure 3. Detailed architecture of Bus Package.

4.2 Vehicle Package

The goal of this package is estimating daily number of trips by vehicles in the city of Tartu. This estimation relies on the data of AVC and Thinnect sensor devices located inside and in the border of the city (Fig. 4). In addition, to the number of vehicle trips, the hourly district level OD-matrix of vehicles' mobility is estimated.

4.2.1 Algorithm

Most cities employ traffic counters to understand the traffic volume in different city regions. However, the counters do not cover all the city streets, and the sensors' counts can be considered a sample of the traffic dynamic. Fig. 4 shows the sparsity of the sensors in the city of Tartu. The standard approach for estimating the absolute number of trips is simulating the urban trips by a traffic simulator. In this work, Simulation of Urban Mobility (SUMO) is used. SUMO is an open-source, microscopic, and continuous traffic simulation developed by the German Aerospace Center and community users. In particular, we used SUMO simulator to overcome the gap between the sensors' traffic counts and actual traffic flow.

In the first step, the map of the region is extracted from OSM [2] and converted to a SUMO network. For the initialization, a set of trips for the whole 24 hours is required. This set of trips consists of two parts. The first part comes from the OD-matrix of bus trips (output of bus package), which gives an initial estimation for trips inside the city. The second part covers the trips into and outside of the city. For this, the data of the AVC sensors and districts populations are used. Lastly, sensors' data is used to calibrate the number of trips. The architecture of vehicle package is

[2]Open Street Map https://www.openstreetmap.org

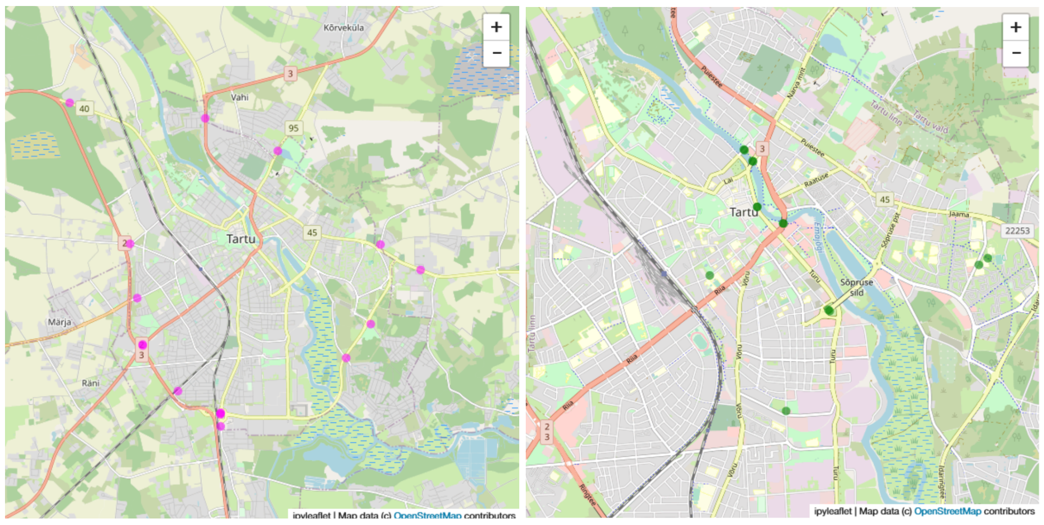

Kõrveküla
Vahi
Tartu
Märja
Räni
Tartu
Sõpruse sild
ipyleaflet | Map data (c) OpenStreetMap contributors
ipyleaflet | Map data (c) OpenStreetMap contributors

Figure 4. Locations of AVC Vehicle counters (left), and Thinnect counters (right).

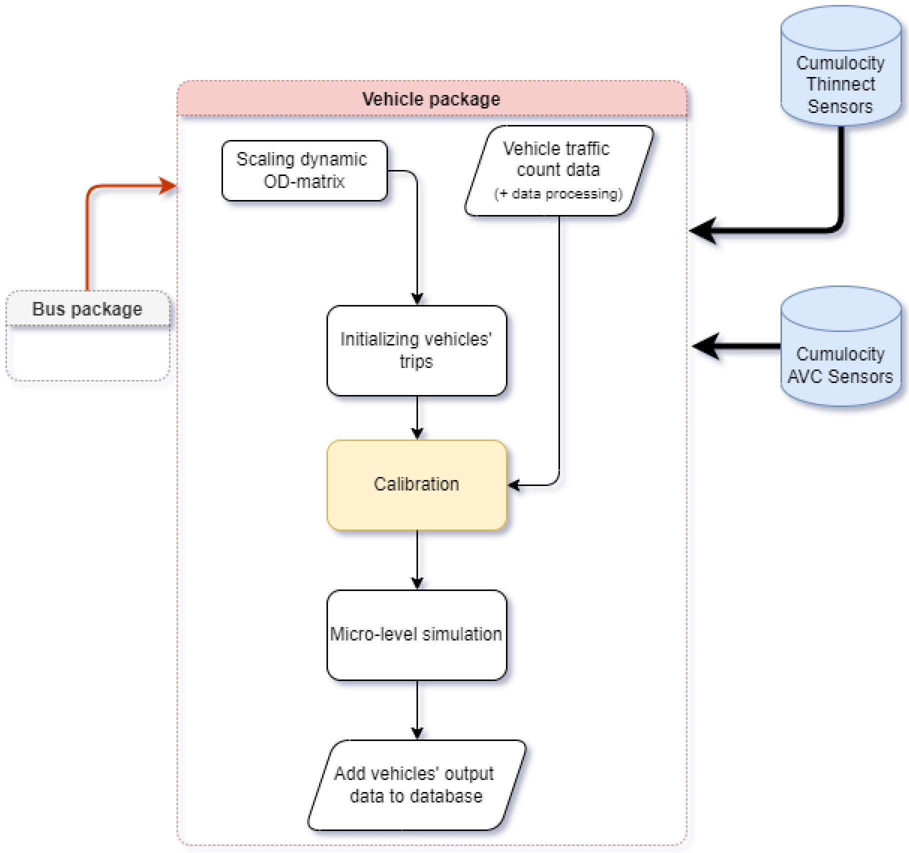

Vehicle package
Scaling dynamic OD-matrix
Vehicle traffic count data (+ data processing)
Initializing vehicles' trips
Calibration
Micro-level simulation
Add vehicles' output data to database
Bus package
Cumulocity Thinnect Sensors
Cumulocity AVC Sensors

Figure 5. Detailed architecture of the vehicle package.

presented in Fig. 5.

4.2.2 Lab Validation

In order to understand how accurately the simulator is estimating the traffic in the city, we run a test in the lab environment for a set of random synthetic trips. In other words, the package is fed by the synthetic sensors' data of vehicle trips as an input; then, the output is compared against the synthetic trips. The lab validation for the vehicle package consists of the following steps.

Validation steps:

1. The schedule of buses is given to the simulator.
2. A set of random vehicle trips are generated. In this step, a trip is a pair of (origin, destination).
3. A set of synthetic multi-modal trips (bus and walk) is generated.
4. Trips are simulated, meaning a route is assigned to the trip from origin to destination.
5. The sensors are located on the map and the trips passed by the sensors are being counted. Therefore, synthetic sensors' data are extracted in this step for the random trips.
6. Synthetic sensors' data from the previous step and the OD-matrix of bus trips (output of the bus package) are given to the algorithm.
7. Finally, the algorithm outputs a set of trips, and the error can be calculated.

A set of 2827 random trips is generated in a sample validation run, and after going through the validation steps, the vehicle package output 2464 tips. For measuring the precision of the method, the relative error is calculated.

$$E = \frac{\text{absolute error}}{\text{real value}} = \frac{|2827 - 2464|}{2464} = 13\%.$$

Fig. 6 presents the hourly distribution of the number of synthetic trips and the estimated number of trips by SUMO after calibration. It can be observed that the error is high from midnight until 5:00 am. The reason for that is the unavailability of bus data during these hours for initializing the algorithm.

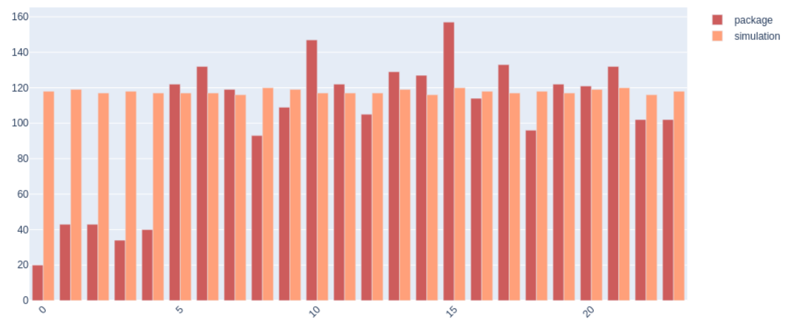


Figure 6. Comparison of hourly number of generated trips and the number of estimated trips by vehicle package.

However, if we only compare the output of the algorithm with the trip data after 5:00 am, as it is presented in Fig. 7, starting from 2238 random trips, the algorithm estimates 2284 trips. The following calculation shows a considerable improvement in error.

$$E = \frac{|2238 - 2284|}{2238} = 2\%.$$

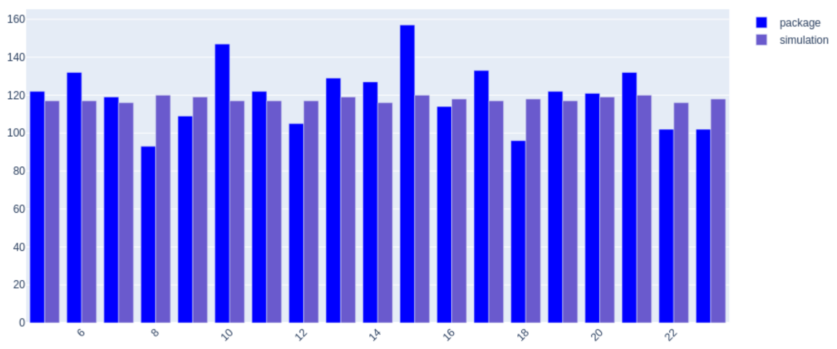


Figure 7. Comparison of hourly number of generated trips and the number of estimated trips by vehicle package, starting from 5 am.

4.3 Bike Package

The goal of this package is to estimate the number of trips by bicycles for every hour, using the Tartu Smart Bike trips as a basis and scaling them to acquire the total amount of bike trips.

4.3.1 Algorithm

The bike package algorithm consists of three main steps. In the first step, the Tartu Smart Bike share data is read from the API. Next, the scaling ratio is calculated. As the last step, the parsing and scaling of the data is conducted.

Data acquisition. The bike package is heavily based on the city bike data. Thus, the first step is pulling the relevant data from the Bewegen API. This is done in batches of 50 trips until all of the trip records for a given date are acquired.

Calculating the scaling ratio. While the Bewegen API gives us the exact number of city bike trips, we also need to find the total number of bike trips, including personal bicycles. For that, a scaling ratio is calculated for a given date and used as a multiplier for the city bike trips. For the scaling ratio, the sensors located on bridges are used (Fig. 8). There are five Thinnect sensors, counting all the “light road users” (*kergliiklejad*), and one ECO sensor located at Turu sild, that counts the cyclists and pedestrians separately. The Turu sild ECO sensor readings are used to calculate the ratio between cyclists and pedestrians crossing the bridge. This ratio is used to acquire the number of cyclists crossing other bridges. Next, the city bike trips crossing any of the bridges are counted, following a simple logic; if the origin and destination docs are on the different sides of the river, the bike has to pass one of the bridges and be included in the sensors counts. Using the total number of cyclists counted at all the bridges and the number of Tartu Smart Bike users crossing any of the bridges, the final share of city bike users (out of total bike users) can be calculated.

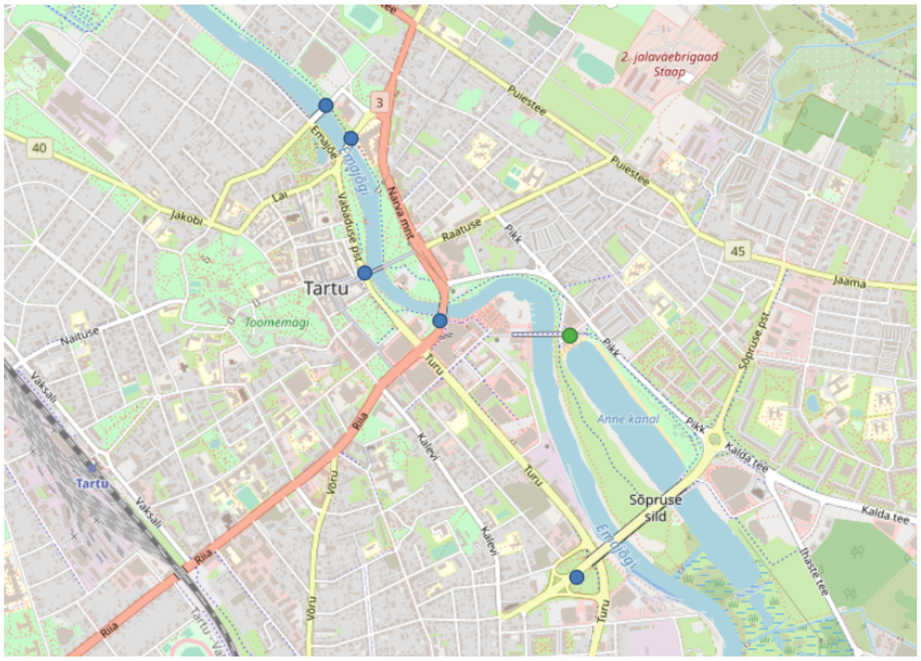


Figure 8. Sensors used for the scaling solution in bicycle package. Blue ones are Thinnect sensors, green is ECO sensor.

Parsing and scaling. As the last step, the city bike data is parsed and scaled. Since the final solution is district-based, the station-level city bike data needs to be mapped to district-level data. During this step, two missing districts are added - as Variku and Supilinna districts do not have any Tartu Smart Bike share stations, the trips from the stations located close to the borders of these two districts are split between given districts. If a station is closer than 100 meters from either of the missing districts, we assume that some of the trips are set out to the corresponding missing district. Once the mapping process is completed, the scaling factor is applied, and the data is saved as an OD-matrix.

4.3.2 Lab Validation

For the lab validation, a synthetic dataset was generated using a traffic simulator, similarly to the vehicle package. Since the Tartu Smart Bike data is the basis of the package and always assumed reliable, the part of the bike package that needed validating, was the scaling. Three relevant datasets were returned from the simulation: city bike trips, Turu sild ECO sensor data and Thinnect bridge sensors' data.

The total of 2145 pedestrian trips, 1680 city bike trips, and 1391 normal bike trips are generated and simulated. Then, the synthetic sensor reading for bridges sensors are extracted from the simulator and the data are fed into the package.

Knowing the actual counts of personal and city bikes in the simulation, the result from the scaling solution could be compared to the actual bicycle count from simulation.

The scaling solution yielded the percentage of 52.02% (*city bikes / total bikes*) on the synthetic data. Knowing the number of city bike trips, the total estimated number of bikes could be calculated:

$$(1680/52.02) \times 100 = 3229$$

Using the total number of bicycle trips from the generated data ($1680 + 1391 = 3071$), the relative error margin of the solution is calculated as follows.

$$E = \frac{|3229 - 3071|}{3071} = 5.16\%$$

4.4 Pedestrians Package

Calculating the number of trips on foot is the most challenging task in modal share calculations. The common approach in the literature for estimating the share of pedestrians is using surveys' data for the whole estimation or as the base of estimation. However, the surveys usually cover a small fraction of the population and have low time-frequency. In the absence of survey data for Tartu city, we developed a novel data-driven approach for estimating the daily number of foot trips in the city, using the sparse pedestrian counters (Fig. 9) and assessing the topology of the walking network. Moreover, we also deliver an hourly district-level OD-matrix of pedestrians. the architecture of pedestrian package is presented in the Fig. 10.

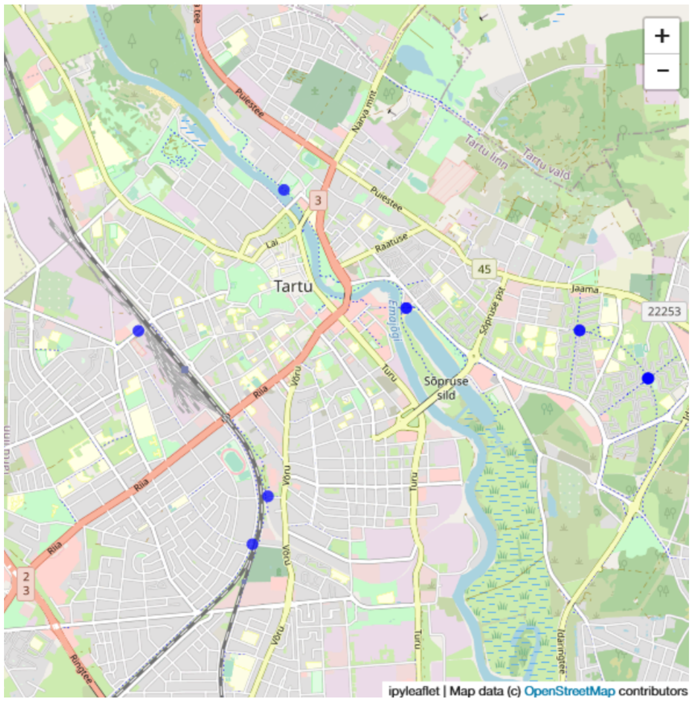


Figure 9. The location of ECO sensors for pedestrian counts.

4.4.1 Algorithm

For every pair of districts i and j, we computed the probability that a trip from i to j passes the sensor k. denote this probability by $a_{ij}^{(k)}$. This probability is computed based on generating and simulating a large number of random trips between different city districts and observing the share of the trips that pass the sensor locations. This observation is highly affected by the topology of the network and its connectivity.

We start with the static bus OD-matrix. Then a set of random trips is being generated following the bus OD-matrix. In order to stay realistic, very long trips are being removed from the set of trips. As a result, all trip distances are less than 3000 meters. An initial OD-matrix $\bar{X}$ is being generated based on the aforementioned OD-matrix and the sensor's data.

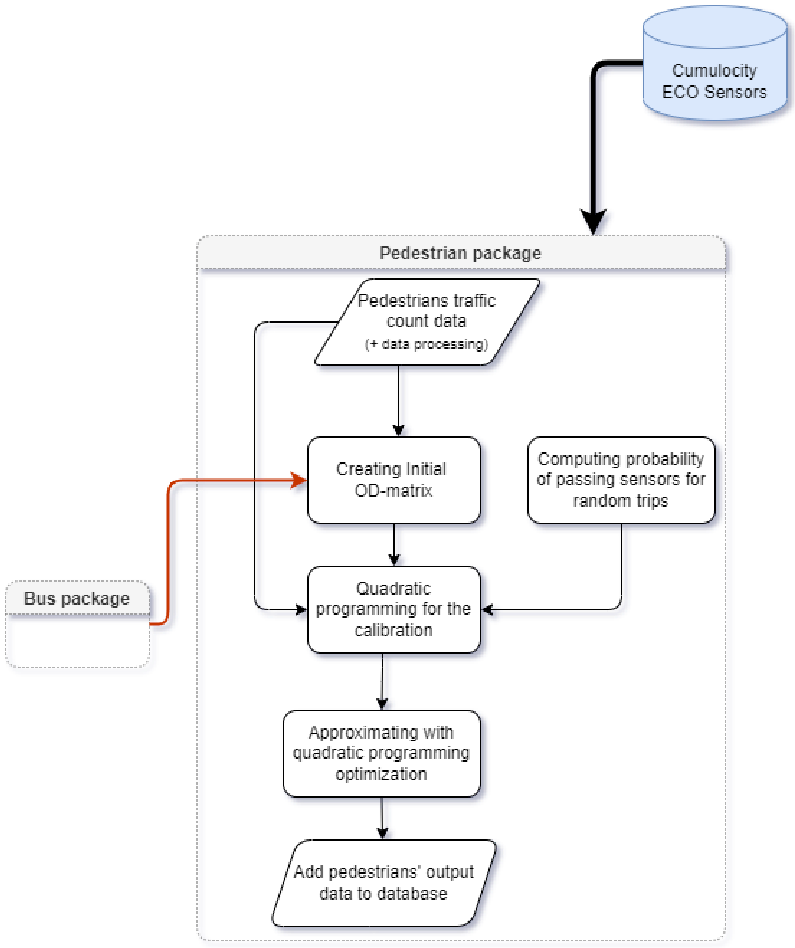

Cumulocity ECO Sensors
Pedestrian package
Pedestrians traffic count data (+ data processing)
Creating Initial OD-matrix
Computing probability of passing sensors for random trips
Bus package
Quadratic programming for the calibration
Approximating with quadratic programming optimization
Add pedestrians' output data to database

Figure 10. The Architecture of pedestrian package.

The objective is estimating the number of trips between each pair of districts, such that the trips that pass through the sensors' locations matches the actual sensor reading. In other words, for every sensor k,

$$\sum_{ij} a_{ij}^{(k)} x_{ij} = y^{(k)}$$

when $y^{(k)}$ is the number of pedestrians detected by sensor k.

At the same time, we want an OD-matrix which is realistic, meaning that does not deviate very much from the initial OD-matrix $\bar{X}$. To ensure this, we are interested to minimize $\|X - \bar{X}\|_2$. This gives us the following quadratic programming problem with

$$\begin{array}{ll} \min & \|X - \bar{X}\|_2 \\ \text{subject to} & \sum_{ij} a_{ij}^{(k)} x_{ij} = y^{(k)}, k = 1, \ldots, \text{number of sensors},\ i, j = 1, \ldots, \text{number of districts}. \end{array}$$

For solving the minimization problem, the optimization package QP Solvers for Python is used. Fig. 11 shows the result for one day of data (hourly OD-matrices are aggregated).

- Green bars are estimated hourly number of pedestrian trips, meaning the sum of all the entries of the final OD-matrix.
- Yellow bars correspond to the expected number of trips passed by the sensors, if random trips are generated based on the OD-matrix X; $\sum_{ijk} a_{ij}^{(k)} x_{ij}$.
- Orange bars are the real sensors' read, $\sum_k y^{(k)}$, in the day of study.

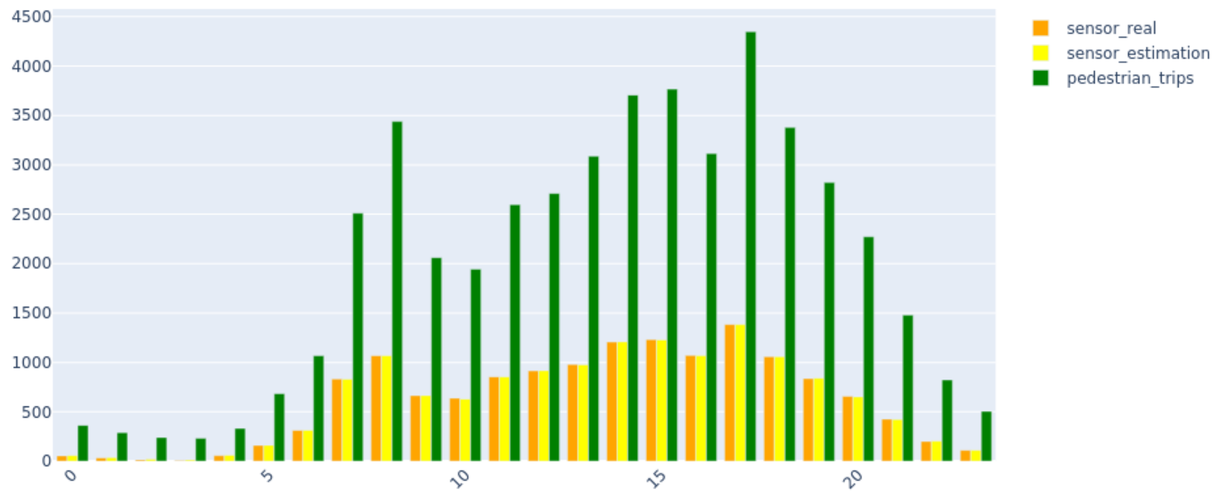
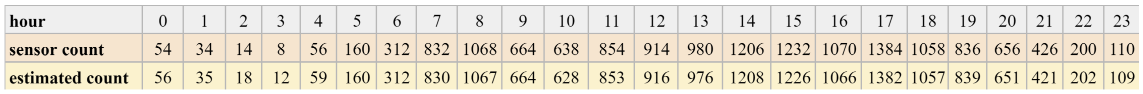


Figure 11. Estimated number of pedestrians' trips, the estimated number of sensor readings, and the real number of sensor readings in different hours of 9 Dec 2021.

4.4.2 Validation

For validating the results of pedestrians' trips, we compare the hourly sensor readings for each sensor, with the estimated sensor reading computed by the algorithm, in the corresponding hour. Fig. 12 presents the aggregated sensors readings and estimates for different hours for 9 Dec 2021.



hour	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
sensor count	54	34	14	8	56	160	312	832	1068	664	638	854	914	980	1206	1232	1070	1384	1058	836	656	426	200	110
estimated count	56	35	18	12	59	160	312	830	1067	664	628	853	916	976	1208	1226	1066	1382	1057	839	651	421	202	109

Figure 12. Total number of sensor readings, and sensors estimations in different hours of 9 Dec 2021.

If we run a simulation based on the The estimated numbers correspond to the the expected number of pedestrians that pass the sensors. Let $S_{h,k}^{(r)}$ be the reading of sensor k during hour h, and $S_{h,k}^{(e)}$ be the estimated reading of sensor k in hour h, then the error is calculated as follows for a selected day.

Sensor name	Sensor count	Estimated count
Anne	1866	1889
Kroonuaua	982	966
Moisavahe	2626	2603
Naituse	3486	3477
Raudtee	260	262
Turu sild	5016	5009
Vaike kaar	530	550

Table 2. Aggregated daily sum of sensor readings and estimations for 9 Dec 2021.

$$Error = \frac{\sqrt{\sum_{h,k}(S_{h,k}^{(r)} - S_{h,k}^{(e)})^2}}{\sqrt{\sum_{h,k}(S_{h,k}^{(r)})^2}} = 1\%.$$

5 GRAPHICAL USER INTERFACE

As mentioned in section 2, users can see the data analysis pipeline output using a web-based dashboard application. Users can browse different dates' results, see how the given date compares to some historical averages and download the OD-matrices data.

5.1 GUI architecture

The server-side of the web application is based on the *Express* framework for *Node.js*. In addition to serving the *HTML/Javascript*-based front-end side of the application, the back-end provides API end-points to first, get the hourly OD-matrices for each modality for a given argument date; and second, query a set of 7-day aggregate statistics for a given argument date. The responses for both cases are structured as JSON objects. The 7-day aggregates are further explained below.

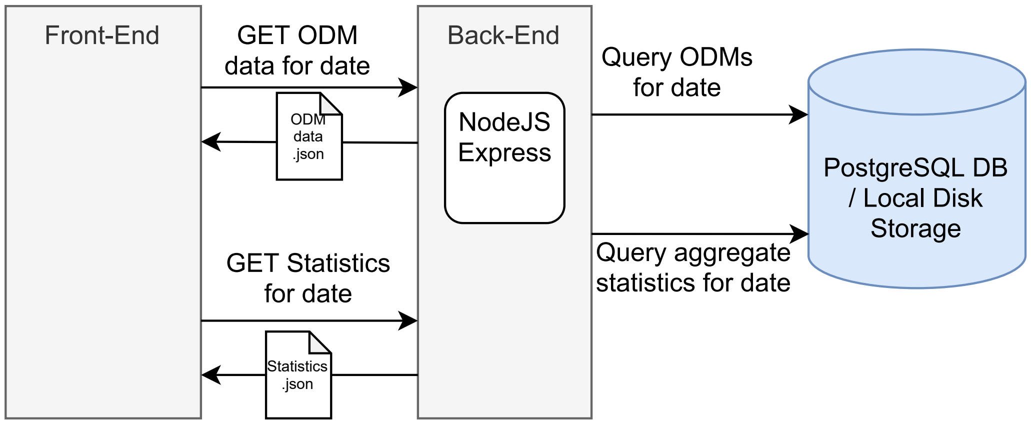


Figure 13. Interactions between web dashboard components and persistent storage.

The interaction between the front-end, back-end and persistent storage of the dashboard is illustrated on Fig. 13.

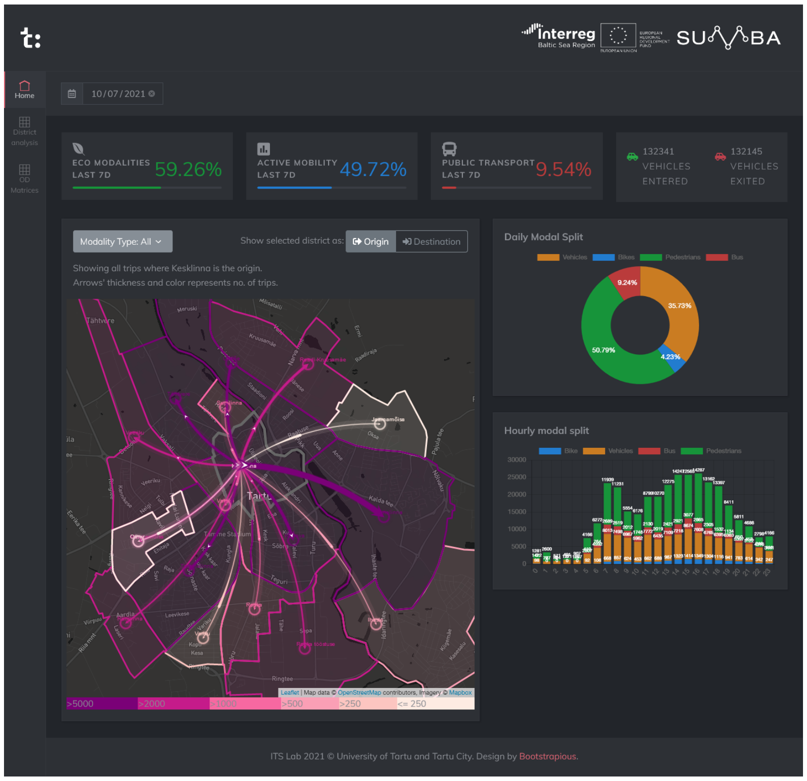
5.2 Functionalities

The graphical user interface consists of three distinct views: “Home”, “District Analysis” and “OD-Matrices”. Each view focuses on the results of a fixed date, and the user can choose other dates.

5.2.1 “Home” View

Home view is loaded by default upon opening the dashboard. A screenshot of the Home view can be seen on Fig. 14. It provides

- a pie-chart, displaying the overall share of each modality for the entire day as a percentage in the city.
- a stacked bar-chart, which displays the absolute numeric values of the modalities for each hour in whole city.
- four numerical statistics. The first three show the average share of modalities for the previous 7 days. These modalities are: “Eco modalities” - the share of bicycles, pedestrians, and bus trips combined; “Active Mobility” - the share of bicycle and pedestrian trips combined; “Public transport”- the share of bus trips. The fourth statistic shows the total number of vehicle trips that were entered the city and the number of vehicles that exited the city (as counted by AVC data) for the current date.
- an interactive map that visualizes the number of trips to a selected district, for all modalities or for a selected modality. The user can also toggle between seeing the number of trips that originated from the selected district and seeing the trips destined to the selected district.



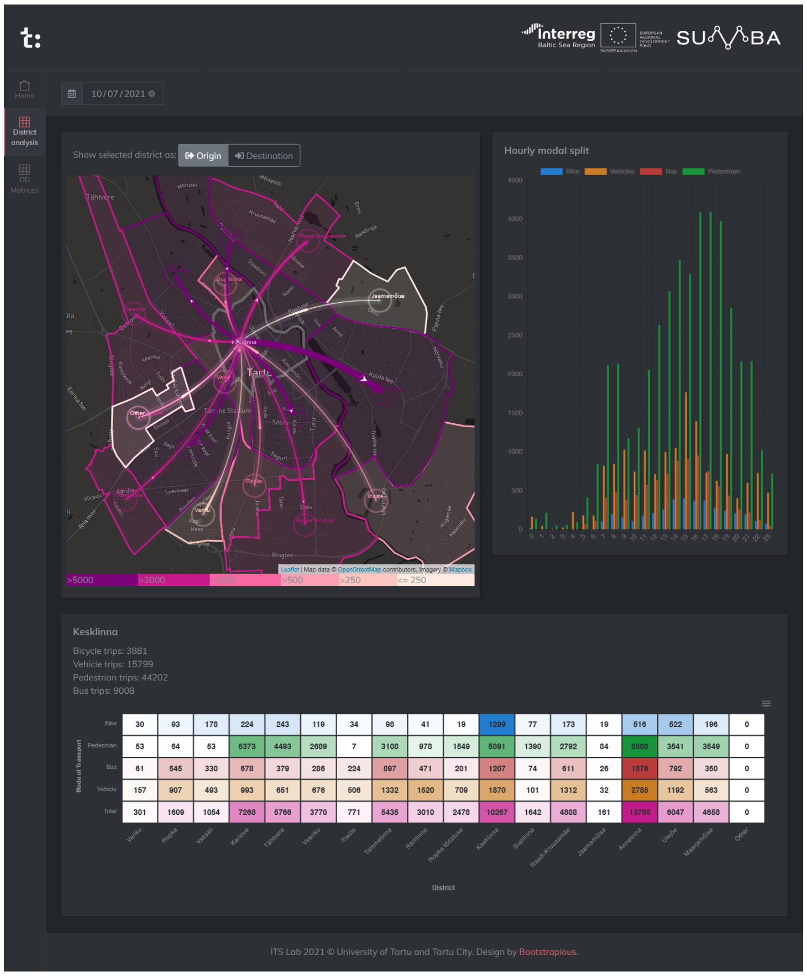
Showing all trips where Kesklinna is the origin.
Arrows' thickness and color represents no. of trips.

ITS Lab 2021 © University of Tartu and Tartu City. Design by Bootstrapious.

Figure 14. The default Home view of the dashboard.

5.2.2 "District Analysis" View

While the Home view mainly showed city-level information, the *District Analysis* (Fig. 15) shows information for a single selected district and selected direction - using the selected district either as the origin or as the destination of trips.



Kesklinna

Bicycle trips: 3881
Vehicle trips: 15799
Pedestrian trips: 44202
Bus trips: 9008

Mode of Transport	Variku	Ropka	Vaksali	Karlova	Tähtvere	Veeriku	Ihaste	Tammelinna	Ränilinna	Ropka tööstuse	Kesklinna	Supilinna	Raadi-Kruusamäe	Jaamamõisa	Annelinna	Ülejõe	Maarjamõisa	Other
Bike	30	93	178	224	243	119	34	98	41	19	1299	77	173	19	516	522	196	0
Pedestrian	53	64	53	5373	4493	2689	7	3108	978	1549	5891	1390	2792	84	8588	3541	3549	0
Bus	61	545	330	678	379	286	224	897	471	201	1207	74	611	26	1876	792	350	0
Vehicle	157	907	493	993	651	676	506	1332	1520	709	1870	101	1312	32	2785	1192	563	0
Total	301	1609	1054	7268	5766	3770	771	5435	3010	2478	10267	1642	4888	161	13765	6047	4658	0

District

ITS Lab 2021 © University of Tartu and Tartu City. Design by Bootstrapious.

Figure 15. The District Analysis view, with detailed hourly modality data for the selected district, and direction.

This view shows an hourly stacked bar-chart indicating the absolute number of trips per modality for the selected district, and direction. The user can use an interactive map to choose the district, similar to the Home view. In addition to the bar-chart, this view displays an OD-matrix-style table showing the numeric values for each district. For example, if the user selects Kesklinna district and direction Origin, the table displays how many trips originated from Kesklinna to the other districts, for each modality, on the selected date.

5.2.3 "OD-Matrices" View

The third view displays daily OD-matrices for the selected date. The view includes five OD-matrices; the first four matrices present the number of trips between different districts separated by each mode of transport, and the fifth OD-matrix shows the aggregated number of trips between districts. Further The user has the option to download each of the OD-matrices as a CSV file, or as an image. A screenshot of this view can be seen on Fig. 16.

Figure 16. OD-Matrices view, which allows downloading the OD-matrices data.

6 FIELD VALIDATION

6.1 Plan Overview

The system is designed to capture large-scale mobility in the city for a whole day. Idealistically, it will be essential to conduct large-scale testing by collecting reference data about the traffic modality in the whole city. However, with the available resources, it is not feasible; therefore, based on the support from the city of Tartu, we planned our testing in the district of Supilinn, which is one of the smallest districts in the city with controllable entrances/exits. We only focused on pedestrians, bikes, and vehicles for the test, since due to electronic ticket validation, the number of bus trips is considered accurate.

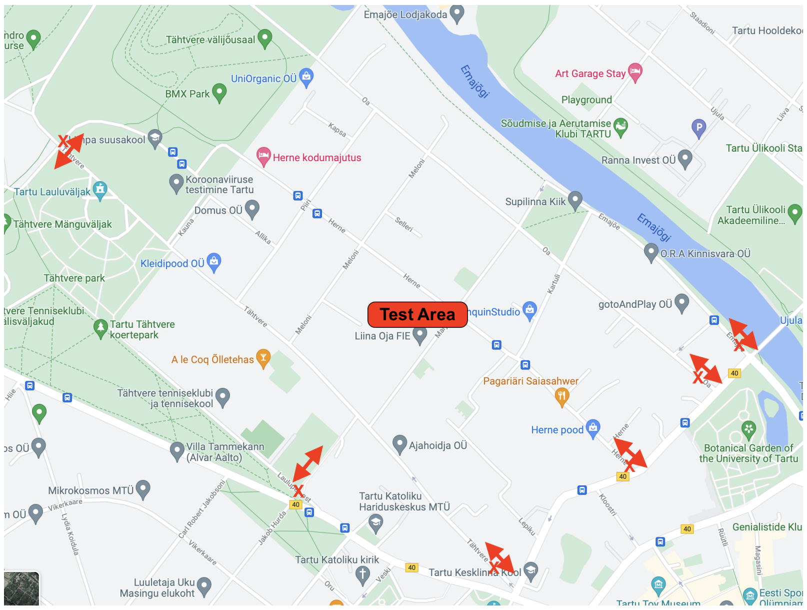


Figure 17. Testing area in the district of Supilinn.

Fig. 17 shows the district as well as the designated spots for doing the manual counts. After manual counts are obtained for different modes of transport, we compared them with respect to the calculated ones using our modal split approach for the districts of Supilinn during the same time interval.

6.2 Results

Table 3 presents the actual and estimated number of trips with different modes in Supilinn on 10 Dec 2021, from 08:00 to 10:00. It can be observed that the result of vehicle estimation differs the most from the actual counts. Two critical factors influenced the system estimation for vehicle count; first, Thinnect sensors did not send data and the traffic simulator relied only on AVC sensors. However, for understanding the traffic flow inside the city, the data of these sensors are crucial. The second factor is the flow of the vehicles that pass through Supilinn, while Supilinn is not their origin or destination. These trips are not considered in our estimated OD-matrix; however, they are counted twice in the real counts, once entering the districts and once exiting the district.

The estimation of bike trips is also affected by the absence of Thinnect data. Moreover, due to the weather condition and the amount of snow, the number of cyclists has dropped to near zero. Given the circumstances, we believe the quality of bike trip estimation is acceptable.

As it is mentioned in section 4.4.2, the pedestrian estimation relies on a novel optimization approach and reflects a very low error rate both in the lab validation and the field testing.

Mode	Real count	Estimated count
Bike	18	10
Pedestrian	338	333
Vehicle	1074	513

Table 3. The number of trips obtained from the field test and the estimated number of trips for Supilinn, on 10 Dec 2021, from 8:00 to 10:00.

Mode	Mod-split real	Mod-split estimation Supilinn	Mod-split estimation Tartu
Bike	1.2 %	1.1 %	0.4%
Pedestrian	23.6%	38.9%	23.5%
Vehicle	75.1%	59.9%	76%

Table 4. Modal split for the real counts in Supilinn, estimated counts for Supilinn, and estimated counts for the city of Tartu on 10 Dec 2021, from 8:00 to 10:00.

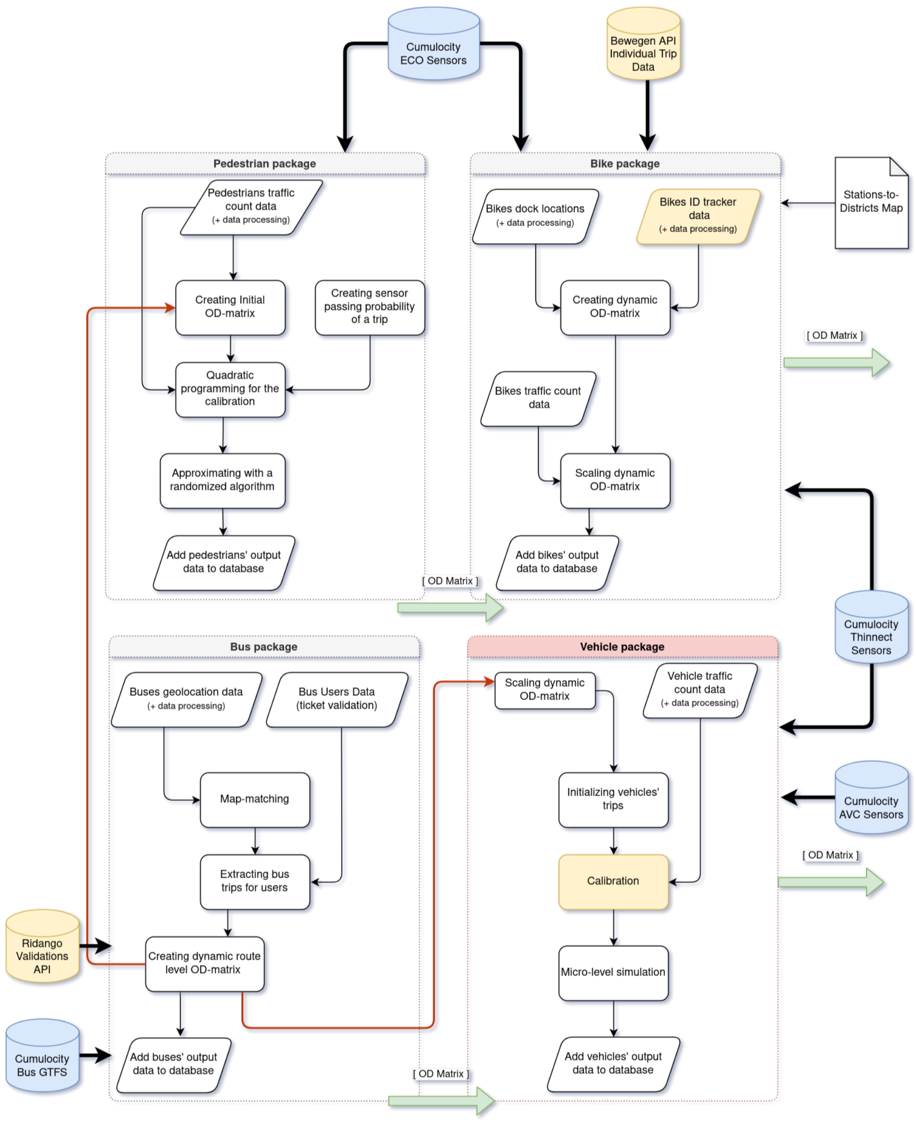
For validating the designed system against the actual field counts, we compare the modal split obtained by the real counts, Supilinn trips and Tartu trips, in the same two hours time interval. The relative error of the modal split for real counts and Supilinn trips equals 27%.

In another approach, we may consider the mobility in Supilinn as a sample of mobility in the city and compare the modal split from the real counts and city-level estimation in the same time interval. With this vision, the relative error of 2% is obtained.

7 RECOMMENDATIONS

For good estimation and performance of our system, it is important to maintain the reliability of the data streams transfer and their quality. Furthermore, to increase the accuracy in estimating the pedestrians and cyclists, we suggest introducing more sensors capable of splitting between cyclists and pedestrians, similar to Turu bridge sensor (ECO sensors). In general, increasing and diversifying the sensors technology or IoT devices can positively impact the system, but they have to be implemented in key/strategic geo-locations. For example, covering all the bridges is essential and also all the entries/exits to the city.

8 APPENDIX



Cumulocity ECO Sensors
Bewegen API Individual Trip Data
Pedestrian package
Pedestrians traffic count data (+ data processing)
Creating Initial OD-matrix
Creating sensor passing probability of a trip
Quadratic programming for the calibration
Approximating with a randomized algorithm
Add pedestrians' output data to database
Bike package
Bikes dock locations (+ data processing)
Bikes ID tracker data (+ data processing)
Stations-to-Districts Map
Creating dynamic OD-matrix
[OD Matrix]
Bikes traffic count data
Scaling dynamic OD-matrix
Add bikes' output data to database
[OD Matrix]
Cumulocity Thinnect Sensors
Bus package
Buses geolocation data (+ data processing)
Bus Users Data (ticket validation)
Map-matching
Extracting bus trips for users
Ridango Validations API
Creating dynamic route level OD-matrix
Cumulocity Bus GTFS
Add buses' output data to database
[OD Matrix]
Vehicle package
Scaling dynamic OD-matrix
Vehicle traffic count data (+ data processing)
Initializing vehicles' trips
Cumulocity AVC Sensors
[OD Matrix]
Calibration
Micro-level simulation
Add vehicles' output data to database

Figure 18. The detailed architecture of modal split.