ÉCOLE POLYTECHNIQUE FÉDÉRALE DE LAUSANNE

ANALYZING MIXED URBAN TRAFFIC BY LINKING LARGE SCALE TRAJECTORY DATASET TO UNDERLYING NETWORK

EPFL MASTERS PROJECT FOR EXCHANGE STUDENTS

LANDTMETERS JOACHIM

SUPERVISOR: PROF. GEROLIMINIS NIKOLAS
CO-SUPERVISOR: BARMPOUNAKIS EMMANOUIL

Academic Year: 2019-2020

CHAPTER 3

EXTRACTION OF MACROSCOPIC TRAFFIC CHARACTERISTICS

This chapter explains the framework of processing the trajectory dataset to get the macroscopic traffic characteristics and perform an elaborate traffic analysis afterwards. First the acquired dataset of the pNEUMA experiment is discussed, afterwards the framework of the developed tool is presented and the last part evaluates the tool for the considered dataset.

The unique dataset used in this research presents enormous opportunities to describe a large-scale network in more detail than ever before. As was explained in the previous chapter, current data sources contain inaccuracies with respect to the traffic flow characteristics and can cause a lot of scatter. The availability of traffic data of every vehicle in the network makes it possible to get the macroscopic parameters for every road with high accuracy. This is the real-life equivalent of the ground truth data in network simulations, since all OD's and routes through the network are known.

3.1 pNEUMA EXPERIMENT

As mentioned in the introduction the pNEUMA experiment (Barmpounakis and Geroliminis, 2020) acquired a large-scale video-image dataset by using ten UAVs during 2.5 hours in the morning peak for all weekdays for the city of Athens, Greece. The dataset is a first-of-its-kind because of the wide scale. A complete trajectory dataset contains an enormous amount of information for the microscopic and macroscopic level because vehicles' movements can accurately be analyzed and less assumptions are needed to measure different characteristics.

Figure 3.1 shows the flight periods of every UAV and the examined area in Athens. The area spans $1.3\ km^2$ with over 100 km of roads, 100 busy intersections and more than 30 bus stops. With computer-vision software all vehicles are tracked and a distinct datapoint is created for every second, containing its position and multiple other variables. Urban traffic is heavily heterogeneous with different vehicle types present in the same spatial environment. In this dataset six vehicle types are distinguished: cars, taxis, buses, powered two-wheelers (PTWs or motorcycles), medium and heavy vehicles.

The collected data consists for every individual trajectory of eight fixed columns and six variable columns with values for every time stamp, table 3.1 lists all these columns. The fixed columns are only labeled or measured once for the whole length of the trajectory. Afterwards a column giving the bearing of every vehicle is added using two consecutive data points, giving the direction of travel for that particular point.

(A) (B)

FIGURE 3.1
pNEUMA experiment: A. time periods of UAV data collection for weekday B. area of Athens, Greece with zones for every UAV (Barmpounakis and Geroliminis, 2020)

These trajectories are mapped in WGS-84 coordinates and there is no link with the underlying network so far.

Fixed	Variable
Tracked vehicle	Latitude
Type	Longitude
Entry gate	Speed
Entry time	Tan. acceleration
Exit gate	Lat. acceleration
Exit time	Time
Travelled distance	Bearing
Average speed	

TABLE 3.1
Columns of collected dataset

3.2 FRAMEWORK

This section presents the main work flow to get the traffic characteristics out of the trajectory dataset. In chapter 2 an overview of research in FDs and MFDs explains the different aspects needed to come to accurate and reliable traffic control. Extracting the traffic characteristics out of the trajectory dataset to estimate link FDs and afterwards the MFD for the studied region forms the scope of this research. The data points (latitude-longitude) are not linked yet with the road network, therefore, a matching of the trajectories to the network to know where every vehicle is located at time t is needed. Figure 3.2 shows the flow chart of the process to get from rough data to traffic flow characteristics for every used edge in the network.

The main steps depicted in the flow chart are:

A Create network graph

B Map match trajectories

C Install virtual detectors

D Calculate traffic characteristics

Each of these processes is explained in more detail hereafter.

3.2.1 CREATING GRAPH

Since the coordinates of the collected data points are not linked to a network topology, it is not known on which specific road a vehicle is driving. A network graph is needed to match every trajectory data point to a physical road. In recent years GIS applications made it easier to extract network structures with their corresponding attributes. Since programming is done in python, the OSMnx package (Boeing, 2017) is an interesting tool to extract road networks all around the world. This recent package extracts a map of the network from OpenStreetMap[1] (OSM) data and simplifies the network topology to a valid graph structure of nodes and edges. Roads in OSM are depicted as edges of a graph structure connected to nodes, which represent the physical intersections. Two crossing edges without a node present on the crossing are not connected and going from one to the other is not directly possible. The OSM database has different topology keys, this makes the extraction of only some types of streets possible. OSMnx has some predefined lists of keys to extract specific network types, e.g. all walkable paths, only driveable streets, etc. Adding or removing keys is easily done to extract a customized selection of streets. In order to extract only streets of a specific area, a bounding-box comprising the most extreme coordinates for every direction (north, east, south, west) is defined.

Position	Attributes
osmid edge	length
N1	lanes
latitude N1	oneway
longitude N1	bearing
N2	highway
latitude N2	dbl left
longitude N2	dbl right
geometry	

TABLE 3.2
Columns of combined dataframe of nodes and edges

The constructed graph is a multidigraph, meaning that the graph consists only of directed edges allowing one direction of travel and multiple directed edges in the same direction between two nodes can exist, i.e. different attributes and weights. When travel is allowed in both directions the edge is split in reciprocal directed edges. In order to work in a more structured way with the graph, the nodes and edges are converted to geodataframes[2]. These dataframes consist of rows for every unique node or edge in the network and columns for every attribute associated with it. Joining these two dataframes and selecting the needed attributes results in a more compact dataframe that is used in later steps. Table 3.2 shows the chosen columns of the resulting dataframe, the first column gives the coordinates and IDs of the nodes making up a directed edge, the second one consists of all the chosen attributes for the specific edge. The bearing specifies the direction of the street with respect to the north, measured counter-clockwise in

[1] https://www.openstreetmap.org

[2] Syntax from pandas package: A dataframe with a geometry column

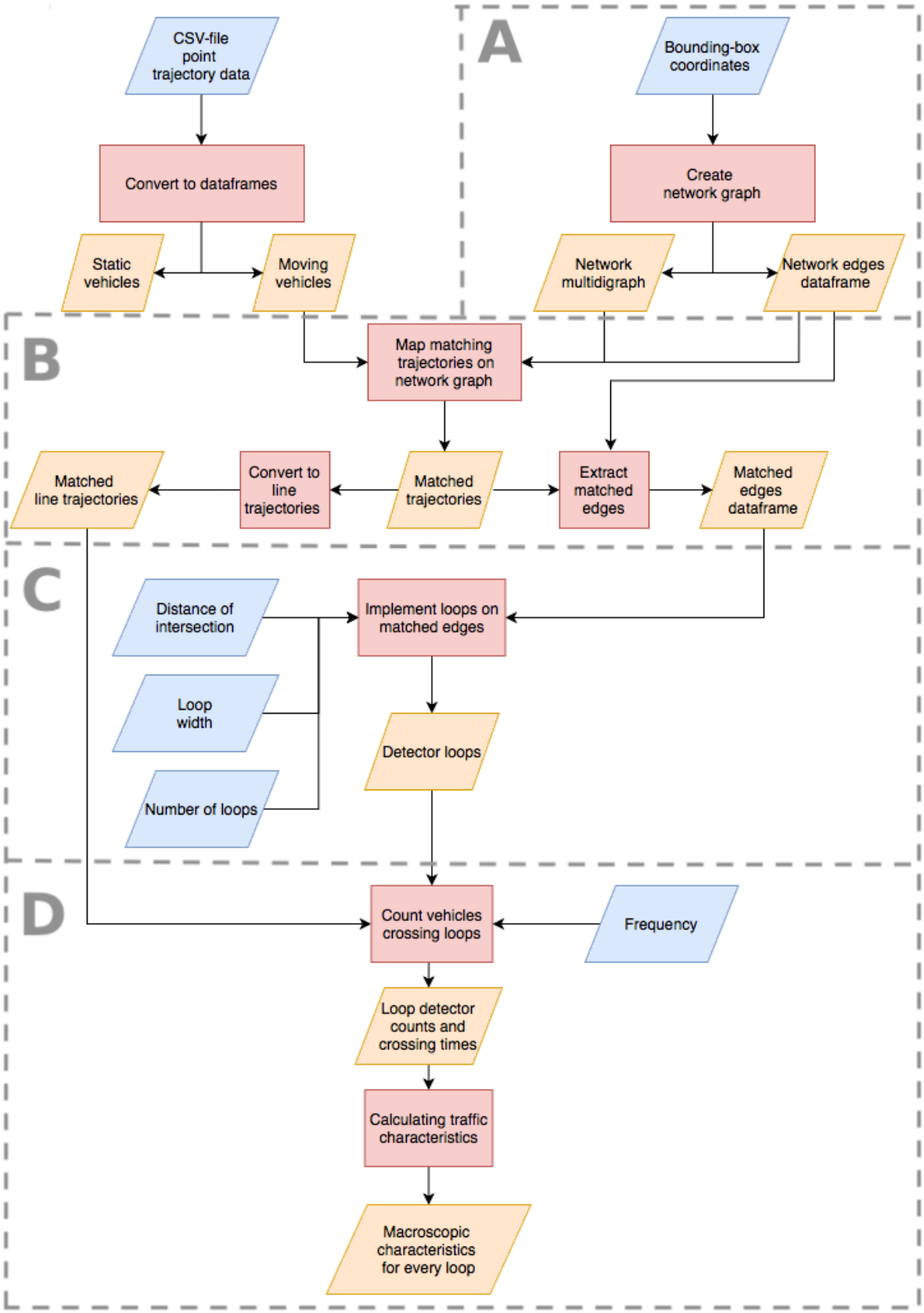

FIGURE 3.2
Program structure

degrees, and is determined with the coordinates of the start and end node. The presence of dedicated bus lanes (DBL) is split up in with flow (right) and contra flow (left) lanes.

Since OSM is a collaborative mapping initiative, updating or adding attributes and streets is very easy, this makes the map very dynamic but also prone to errors. Therefore it is important to keep in mind that the network structure may have changed over time, e.g one-way streets and pedestrian zones. Customizing the keys for the network type and the attribute tags, which can be very interesting for future iterations to extract a more accurate network or research extra attributes, is easily done using OSMnx.

Network type	Nodes	Edges
Drivable	1028	1872
All	2426	5613

TABLE 3.3
Comparison of network types

Figure 3.3 shows the street network and the difference between a chosen network type for Athens in a specified bounding-box. It is clear that extracting all streets results in a very dense network with a lot of edges around intersections, table 3.3 quantifies this difference. Note that the drivable network is a subset from the network containing all streets.

FIGURE 3.3
Street network for different network types

3.2.2 MAP-MATCHING

Knowing on which road a vehicle drives at a specific time gives valuable microscopic traffic information that can be translated to the macroscopic level. In order to achieve this, the acquired trajectory data with a latitude and longitude for every point has to be mapped to its corresponding edge of the underlying road network. In recent years numerous map-matching algorithms have been developed. Ranging from purely geometrical algorithms to probabilistic ones. For this research no new map-matching algorithm is developed, instead a publicly available open-source algorithm for python is chosen that is easily

compatible with the OSMnx package. The *leuvenmapmatching* package (Meert and Verbeke, 2018) fits these requirements. This algorithm is based on the hidden Markov model (HMM) approach presented in (Newson and Krumm, 2009).

3.2.2.1 ALGORITHM BACKGROUND

An HMM is a probabilistic model that is an extension of a Markov model but now the states are hidden and only observations associated with these states are observed. Following Rabiner (1989) an HMM comprises five elements:

1. N hidden states $S_1, .., S_N$
2. A transition model $a_{ij} = P(q_{t+1} = S_j | q_t = S_i)$
3. An initial probability distribution $\pi_i = P(q_1 = S_i)$
4. Domain of possible observations
5. An observation model $b_i = P(o_t | q_t = S_i$

At every time instance the model is in one of the possible states and the transition model defines the probability of the model being in another state at the next time instance given the current state, known as the Markov property. The initial probabilities are necessary to know the state of the model at the first time step. The difference with the standard Markov model is the observation model, since it is not known directly in which state the model is at a given time but only observations are available, the probability of an observation corresponding to a state has to be defined. When all the parameters of the model are defined some inference problems need to be solved, depending on the application. An overview of these problems is given in (Devos, 2018).

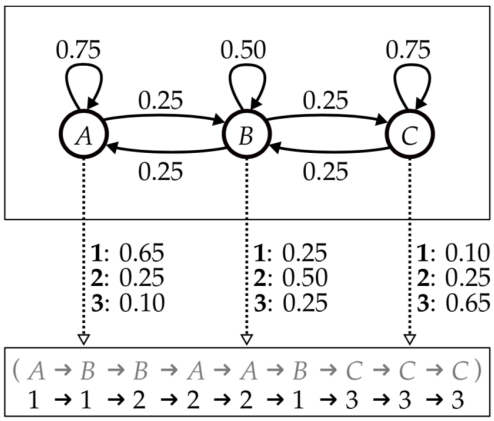

FIGURE 3.4
Example of HMM taken from (Devos, 2018)

In figure 3.4 an example of an HMM is shown. Instead of observing states A, B and C directly, observations 1, 2 and 3 are observed with a certain probability associated with the states. It is said that the states emit the observations with this probability. For this example, the probabilities to observe 1 are 0.65, 0.25, 0.10 for A, B and C respectively.

In Devos (2018) the theoretical principles of the used algorithm are elaborately explained[3]. For map-

[3]For more explanation about their *leuvenmapmatching* algorithm the authors referred to (Devos, 2018)

matching, the desired output is a sequence of states that corresponds to a sequence of observations, this is one of the stated inference problems, referred to as the *best-state-sequence* problem.

$$\underset{q_1,..,q_T}{\operatorname{argmax}} P(q_1, .., q_T | o_1, .., o_T, \theta) \tag{3.1}$$

θ in equation 3.1 depicts the parameters of the HMM. Using dynamic programming this problem is solved by the Viterbi algorithm (Forney, 1973). This algorithm uses recursion to determine the *best-state-sequence* to arrive in state S_j, see equation 3.2 with ξ the Bayes normalization factor as stated in (Devos, 2018).

$$\delta_t(i) = \begin{cases} \frac{1}{\xi_1}\pi_i b_i & t = 1 \\ \frac{1}{\xi_t}\max_i[\delta_{t-1}(i)a_{ij}]b_j & 1 < t < T \end{cases} \tag{3.2}$$

This theoretical formulation is translated to the map-matching case. For equation 3.2 this means that for every edge the best route of edges is calculated to maximize the probability given a sequence of observations.

- edges $(N1, N2)$ form the set of states
- all points of the map are in the domain of observations, representation depends on chosen projection (lat-lon, euclidean). Trajectories and the map thus need to be represented in the same projection.
- the transition model to move from one edge to the next is modeled with an exponential distribution of the difference between the distance of consecutive observations and the corresponding distance along the network edges (Newson and Krumm, 2009)
- the observation model to link an observation to a state with certain probability, modeled with an exponential distribution of the straight-line distance of the observation to the edge (Newson and Krumm, 2009)
- initial probabilities are defined by the observation probability of the first observation of a trajectory

3.2.2.2 IMPLEMENTATION

Firstly, the needed network graph for the algorithm is constructed from the extracted OSM graph by taking the node id, node location (lat-lon) and all the directed edges start and end nodes. Secondly, to map every individual trajectory some parameters have to be specified, namely an initial searching distance and a maximum searching distance. The former sets a searching radius to find all possible starting edges for the first observation, the latter specifies the maximum allowable straight-line distance of the observation to one of the possible edges. Translated to the theoretical model, the set of possible states for the first observation is constructed with the initial maximal distance and subsequently the initial probabilities for these states are calculated for those edges within the maximal allowed distance. It is clear that these two parameters limit the number of possible states for every observation and thus have a great impact on the speed of the algorithm. Since urban networks are quite dense, the number of possible edges can increase rapidly with slightly higher parameter values, it is thus not advisable to set general values for the whole trajectory dataset. By looping over all trajectories, starting with the same initial values for the parameters these parameters can be increased for every trajectory individually if the algorithm cannot find a matching result. The result of matching a trajectory to the network is depicted in figure 3.5.

FIGURE 3.5
Result of map-matching for one trajectory

3.2.3 VIRTUAL LOOPS

At this stage every trajectory has for every data point a matched edge. Extracting macroscopic traffic characteristics for a specific location is achieved through placing virtual loops, referred to as detectors hereafter, over the corresponding edge. These detectors are pairs of parallel straight lines perpendicular to the corresponding edge and should at least cover the whole road width. The flow chart, figure 3.2, states three input variables that are freely chosen, namely a specified distance from the upstream and downstream intersection, the width of a virtual detector pair making up the measurement region (ΔX) and the number of detectors on an individual edge. It should be noted that the intersections of the network graph are point approximations of the real intersection, substantiating the necessity of the 'distance from intersection' (DFI) parameter. When the chosen number of detectors is higher detectors are evenly distributed over the length between the first and last detector.

3.2.3.1 IMPLEMENTATION

From the map-matching result a subset of the street network gives all the edges used by at least one vehicle. This prevents placing virtual loops on edges with no matched vehicles. Since the two distance input parameters are in meters the network is projected to euclidean space to assure right placement of the loops. The first step is finding the right locations of every detector pair on the edge given the three input parameters.

Once the right locations are found, the points are transformed to lines with a length long enough to detect every vehicle matched to the edge and a bearing that is perpendicular to the edge. Afterwards, the dataframe is again converted to the right coordinate reference system (crs). When one of the input parameters makes a right placement impossible, some adjustments are made:

1. **Distance from intersection (DFI)** When the edge is not long enough to assure the specified distance from the intersection, all detectors, depending on the specified number, are placed in the middle of the edge.
2. **Detector width (ΔX)** If an edge is shorter than ΔX, the detector is placed in the middle with a newly specified ΔX relative to the length
3. **Number of detectors too high** Given the number of detectors, an inter-detector length is determined

and if this causes a violation for one of the other two parameters, all detectors are again placed in the middle

The adjustments assure that the ΔX is preserved as long as the edge length allows it, making comparisons possible between different detector locations because the space-time window is the same. Courbon and Leclercq (2011) state that specifying this window is difficult which implies that comparing windows of a different size is inaccurate. As an example figure 3.6 depicts the placement of three detectors per edge.

FIGURE 3.6
Detectors on used edges of network with specified DFI, ΔX and number of detectors per edge

3.2.4 TRAFFIC CHARACTERISTICS MEASUREMENT

These detectors differ from the traditional double loop detectors on roads because the availability of all trajectories makes it possible to use Edie (1963) for every chosen space-time area on every edge by using the crossing times of every individual vehicle trajectory and calculating their time spent (VHT) and distance travelled (VKT) when inside the space-time area.

Until now the trajectories are made up of chronological points and a conversion to line segments for consecutive points is needed. The crossings of these lines with the created detectors give the exact location and time of a vehicle entering and leaving a detector, assuming constant movement between consecutive trajectory points allows the usage of linear interpolation. The equations 3.3, 3.4 and 3.5 give the average flow, density and speed for every detector, respectively.

$$q_i = \frac{\sum_k x_k}{\Delta X \Delta T} \tag{3.3}$$

$$k_i = \frac{\sum_k t_k}{\Delta X \Delta T} \tag{3.4}$$

$$v_i = \frac{q_i}{k_i} \tag{3.5}$$

with x_k, t_k the VKT and VHT of every vehicle k in space-time area $\Delta X \Delta T$, respectively. The average speed is equal to the space-mean speed. In contrast to traditional loop detectors, ΔX can be as large

as preferred, no assumption of constant speed is needed since the VKT of data points of a trajectory in between the detector edges is determined for every time interval ΔT.

3.2.4.1 IMPLEMENTATION

In order to determine the macroscopic traffic characteristics enumerating all the selected edges with their detectors and the trajectories is done for a chosen space-time area. VKT and VHT are calculated for every individual trajectory and the values are assigned to a specific time step.

To prevent negative values, tags are used to assure that VHT and VKT are only measured when the detector is crossed correctly. This tag is also necessary to include VKT if a vehicle's line segment is completely inside the space-time area. After the vehicle crossed the detector, these tags are reset to allow the same vehicle to cross the detector again, i.e. vehicle performing loops in the network.

All the values for one edge are saved in a dataframe with one time step per row and lists in every column comprising every individual trajectory's VHT and VKT for that particular time step. Keeping the individual trajectory values gives more possibilities for further analysis, for example splitting by mode or a travel time analysis based on entry and exit times of specific detectors. For different edges this results in a list of dataframes and additionally a new list of dataframes with the macroscopic value for every time step by using the equations 3.3, 3.4 and 3.5.

3.3 APPLICATION AND EVALUATION

This section describes the process to attain accurate values of the macroscopic traffic characteristics used for later analysis. Applying what was discussed earlier to the dataset and doing some extra iterations results in an eventual street network with an accurate map-matching of all the trajectories.

3.3.1 DATA

The used dataset in this research consists of 10659 different trajectories with the longest trajectories having a total time of fifteen minutes. Leading to a total set of over 1.5 million datapoints. Of these trajectories, 69 did not move during the whole recording period and are therefore not considered anymore. For the remaining dataset the share of every vehicle together with its corresponding dimension, length and width in meters, is shown in table 3.4. Four trajectories are not any of these vehicle types and are not used for further analysis. Cars, PTWs and taxis make up more than ninety percent of all the trajectories.

	Car	Taxi	Bus	PTW	Medium Vehicle	Heavy Vehicle	Total
Share	3989 (38 %)	1739 (16 %)	198 (2 %)	4037 (38 %)	526 (5 %)	97 (1 %)	10586 (100 %)
Dimension (l,w)	5,2	5,2	12.5,4	2.5,1	5.83,2.67	12.5,3.3	

TABLE 3.4
Proportion and dimensions in meters of vehicle types in the dataset

3.3.2 MAP-MATCHING ISSUES

Selecting the right network type is of crucial importance to get accurate results. As mentioned earlier there can exist a high difference in the number of nodes and edges due to the chosen network type. The highly accurate trajectory dataset asks for an accurate enough street network, however choosing a too detailed map will result in long computational time for little added value. For the given dataset the network type should comprise all the roads whereon the recorded vehicles can drive. In the first try, the chosen type contains all drivable roads which results in extracting all the streets for motorized traffic for the selected region. Performing the map-matching algorithm brings forward some issues, as listed below and depicted in figure 3.7.

1. **Too low initial distance** When the maximum initial distance is not high enough it can happen that the desired edge is not in the set of starting states for the first observation. Since the graph is build solely on the location of the nodes, the found edges are those of which the starting node is in the set of possible starting nodes. Therefore, when the starting node is not inside the initial searching area the corresponding directed edge will not be considered when the initial probabilities are determined for this first observation and the point is assigned to the wrong edge.
2. **Too high maximum distance** The straight-line distance from the observation to all the edges within the set of states cannot exceed the maximum distance parameter. When this happens the algorithm will fail to find a corresponding edge for that observation. Therefore the maximum distance should be high enough for all the points along the trajectory but this, on the other hand, can result in incorrect matching.
3. **Missing edges** It can happen that for some trajectories the correct edge is missing in the graph, two sorts of cases were encountered. The first one is due to vehicles driving on the pavement in the opposite direction of the corresponding road, ignoring one-way street signs or using pedestrian streets or some other streets which were not extracted from OSM because they are not part of the chosen network type, see section 3.2.1. The second case consists of roads missing that should be part of the network type. This is mainly observed for locations where a DBL is present.

FIGURE 3.7
Matching issues: a. maximum initial distance, the upper figure (1) has a too low initial distance b. maximum distance, the right figure (2) maps wrongly because of a too high distance c. missing edges, the trajectory is wrongly matched at the beginning and the end. It makes a U-turn on the pavement in the beginning (1) and at the end goes into a pedestrian street (2)

The initial maximum distance is set to the maximum length of an edge in the network to solve the first issue. Assuming that an observation will not be further away from its correct match than this maximum

distance ensures that the right edge is always part of the initial set of possible states. Situation 2 in figure 3.7a shows that the first observations of the trajectory match to the right initial edge.

For the second issue the reason of the incorrect matching shows a possible shortcoming of using the HMM algorithm for this dataset with a one second sample rate. Since the observation probabilities are determined for all the states in the vicinity of the observation, it can happen that the trajectory is closer to the incorrect edge, given the dense street network. When the vehicle stops afterwards in between the two edges, the Viterbi algorithm will define the incorrect matching as the *best-state-sequence* up until the vehicle starts moving again. Because of the Markov property this sequence is not shifted to the correct one since no other route maximizes the inference problem, due to the stopping, and a detour is made. By increasing this distance when the algorithm fails this issue is resolved in an iterative way, although increasing it too slowly will impact the speed of the algorithm, increasing it too fast can again lead to the incorrect matching, especially at dense areas in the network, e.g. intersections. Situation 1 in figure 3.7b now does not make a detour at the intersection. In the comments the authors acknowledge that setting the maximum distance is a hard cut and quite arbitrary and propose using a minimum probability but choosing this value is also arbitrary and less interpretable. Using the maximum distance is thus preferred.

Solving the last issue is split up, given the two encountered cases. When connections are missing that should be part of the network, the open access of OSM allows adding these connections. For this dataset mainly DBLs are missing and are added by changing the respective tag and using this to create an extra edge if it is a contra-flow lane in a one-way street. The bundled bus trajectories show where buses drive and should overlap with the bus lanes, see figure 3.8. The cases where roads are used that are not part of the network, customizing the list of keys and tags can help to extract the best network, e.g. including pedestrian streets in the drivable network. Nevertheless, this only solves the most clear cases, when vehicles are maneuvering and using the pavement or ignoring one-way street directionality changing the network structure is not advisable. Instead a new column is created in the trajectory dataframe containing the difference between the bearing of the trajectory point and the matched edge, quantifying the mismatch.

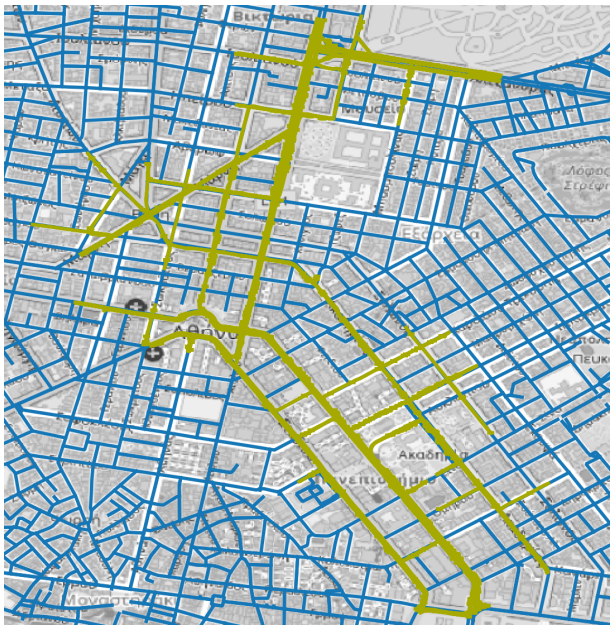

FIGURE 3.8
All the bus trajectories in the researched area

Taking into account the solutions for the issues mentioned, the parameter values for the map-matching algorithm are set. For the initial maximum distance this is the maximum length of all the edges in the

network and the maximum of the straight-line distance to a possible matching edge is set at five meters and increased by five every time the algorithm fails. Five meters seems a reasonable distance knowing the high sample rate of the trajectory dataset and the dense network structure.

3.3.2.1 GRAPH ADJUSTMENTS

After applying the proposed adjustments, the street network is extracted a second time with a customized set of keys, including pedestrian streets, and the DBL tag is used to create missing edges where needed. Figure 3.9 shows the network with the pedestrian streets added to the network type.

FIGURE 3.9
Network with pedestrian streets included

In figure 3.10, the newly added bus lanes are depicted on the left and the total of all DBLs after the addition on the right. These bus lanes can be used by taxis and PTWs, making these missing links not negligible knowing the high shares of these two modes in the dataset.

(A) (B)

FIGURE 3.10
DBL in network: A. Newly added DBL B. All DBL in network

3.3.2.2 EVALUATION

After running the algorithm with the right parameter values, taking into account the encountered issues, the whole trajectory dataset is matched to the network. However, it is not assured that even after addressing these issues every trajectory is correctly matched. Evaluating the matching result is therefore essential in order to be sure that the matched result corresponds with the observations. Since ground truth data is not available, the video footage is not analyzed in this research, some other defined variables can shed more light on the accuracy. In Brakatsoulas et al. (2005) a map-matching algorithm uses the Fréchet distance to match trajectories to a map. This distance gives a quantitative measure of similarity between two curves and is better known as the walking-dog problem: what is the longest length of a leash if a dog walks on one curve and a person on the other without the ability of backtracking. Calculating this distance continuously is computationally expensive and Eiter and Mannila (1994) present a discretized version of this distance, but interpreting this distance on its absolute value is difficult because the discretization seems to give high distances even for well-matched trajectories. Keeping the walking-dog problem in mind, a combination of other variables is used to evaluate the overall map-matching and find the badly matched trajectories. The variables are:

- median straight-line distance
- mean straight-line distance
- maximum straight-line distance
- difference between trajectory and matched length
- bearing difference between trajectory point and matched edge

Figure 3.11 shows the histograms for the different variables after map-matching all the trajectories. These clearly show that the overall accuracy of the map-matching is good, as was expected given the high sample rate of the trajectories and the well-considered setting of the parameters discussed earlier. The median and mean follow the same shape and have almost equal values indicating that for most trajectories possible mismatches are not due to a couple of large outliers. The maximum distance has its peak around ten meters which is reasonable given that road width is not given in the network graph, but it does show a second peak around twenty meters. Overall the bearing difference is very low for a large majority of the trajectories. A high value for any of these variables can point to a mismatch between the trajectory and the network somewhere along the path.

The second peak observed in the maximum distance histogram shows that there still are some possible mismatches between trajectories and the network, although the adjustments of the graph have improved the overall map-matching a lot, see table 3.5 for the comparison of the maximum distance for both network types. The most prominent improvement is the value for the 75th-percentile, with the adjusted network three quarters of the trajectories have a maximum distance lower than ten meters. Figure 3.12d shows that ten meters is a very accurate value because the width of the roads is not taken into account and the location of the edges extracted from OSM may not always be in the middle of the ground truth road, resulting in a systematic deflection.

A deeper look into the trajectories having a higher maximum distance than ten meters gives valuable information about the reasons of mismatches. The reason for the second peak around twenty meters follows from the way the graph of the map-matching algorithm is built. Only the node locations are used to define an edge, omitting any curvature of the edges. Figure 3.12b shows the situation when the edges around Omonoia square are left out, it is clear that the peak disappeared for the trajectories not travelling on one of the Omonoia edges and that for the trajectories passing Omonoia, figure 3.12a, the maximum distance lies around twenty meters.

	Original [m]	Adjusted [m]
mean	10.73	9.18
std	9.51	7.85
min	0.14	0.14
25 %	4.54	4.36
50 %	7.35	6.80
75 %	16.51	9.92
95 %	24.68	24.10
max	122.63	106.96

TABLE 3.5
Comparison of maximum distance after map-matching for the original and adjusted network

Further analyzing the dataset without the trajectories passing Omonoia, the correlation between the evaluation variables provides a clear picture of the variables that can flag a mismatch. In tables 3.6 the correlation matrices are shown for two different selection procedures. The first one, left figure, contains all the trajectories with a maximum distance higher than ten meters. The other one, right figure, has all the trajectories with an average speed higher than one kilometer per hour when their bearing difference exceeds the ninety degrees threshold. The reason for this last selection comes from the fact that vehicles stopping for a longer time have a different bearing because of maneuvers and parking in a direction leading to a high bearing difference with the matched edge. Therefore these vehicles are not mismatches and are left out in the second procedure.

	MD	**LD**	**LDr**	**BD**
MD	1	0.5	0.28	0.26
LD	0.5	1	0.68	0.37
LDr	0.28	0.68	1	0.59
BD	0.26	0.37	0.59	1

	MD	**LD**	**LDr**	**BD**
MD	1	0.53	0.21	0.26
LD	0.53	1	0.63	0.5
LDr	0.21	0.63	1	0.83
BD	0.26	0.5	0.83	1

TABLE 3.6
Correlation matrices for two selection procedures, left: maximum distance, right: average speed (MD: maximum distance, LD: length difference, LDr: relative length difference, BD: bearing difference (>90 deg)

All correlations are positive, as is expected, since high values for any of the variables points out possible mismatches. From these tables the most interesting correlation is the one between the relative length difference and the bearing difference, vehicles traveling in a direction that does not correspond with the direction of the matched edge have a very high difference in their actual and matched path. The reason for these mismatches is due to the third issue explained earlier, see 3.3.2. where vehicles ignore one-way streets or use the pavement to drive in another direction than the orientation of the corresponding road. A majority of those cases are PTWs, they use more frequently the pedestrian streets or other paths connecting two roads, leading to very high length and bearing differences.

In conclusion the algorithm gives accurate results, when mismatches occur it is because some edges are missing in the network, e.g. pavements and other walking paths, or maneuvers and driving behaviour that lead to high bearing differences. A maximum distance of ten meters is the reasonable threshold, excluding those edges with a lot of curvature, and additionally high length differences and bearing differences allow to assume some directionality issue with the matched edges.

3.3.3 TRAFFIC CHARACTERISTICS

After improving the map-matching result, the traffic characteristics are determined for every detector on the edges of the network. Taking into account possible mismatches, the tags prevent from counting crossings in the other direction and trajectories with a bearing difference higher than ninety degrees are skipped, choosing a lower difference is possible but this can wrongfully leave out trajectories due to lane changes, for example.

3.3.3.1 SPACE-TIME AREA

Selecting a window to aggregate all VKT and VHT is an important but also arbitrary task. ΔX can be very short or as long as the edge length, similarly ΔT can take one second steps or be fifteen minutes long. Figure 3.13a shows scatter plots for different ΔT. With only fifteen minutes of available data, taking too long ΔT results in only a few data points, and too short in very scattered data with peaks that are an overestimation of the ground truth. Aggregating the data to a multiple of ΔT is still possible without making any assumptions or incorrect averaging, i.e. time-mean speed instead of space-mean speed, because the individual VKT and VHT are available.

The plots in figure 3.13b show FDs for different ΔX. It is clear that wider loops show more scatter because different traffic states are mixed more frequently. The width for the detector also determines how many detectors per edge can physically be placed. As shortly mentioned in the previous chapter, Courbon and Leclercq (2011) state that placing many detectors in order to measure as many traffic states as possible gives good MFD estimates. Additionally, vehicles that do not cross both edges of the detector are left out for the traffic characteristics calculation, this would happen more frequently with wider detectors.

3.3.3.2 MIXED TRAFFIC

The aggregation of VHT and VKT can also easily be split up by mode, equations 3.6 and 3.7

$$\sum_{j}^{N} t_j = \sum_{m} \sum_{j}^{N_m} t_j \tag{3.6}$$

$$\sum_{j}^{N} x_j = \sum_{m} \sum_{j}^{N_m} x_j \tag{3.7}$$

With m for the different modes and N_m the subset of trajectories for a specific mode. Because temporal changes in the number of vehicles passing influences the magnitude of the flow and density, comparing the space-mean speed between modes is preferred.

3.4 SUMMARY

The presented framework leads to the extraction of traffic characteristics, i.e. flow, density and space-mean speed, for the highly-detailed trajectory dataset of the pNEUMA experiment. In order to know where vehicles are at a particular moment in time, a map-matching algorithm is used to get the most probable edge for the vehicle at every time instance. With detectors placed on every used edge, calculating VKT and VHT to come to the macroscopic traffic characteristics becomes very easy for selected locations in the network. Afterwards some issues are elaborately discussed and resolved, improving the result and

reliability of the map-matching output a lot. For this case the initial searching distance is set equal to the longest edge in the network and the maximum allowed straight-line distance starts at five meters increasing with five when the algorithm fails. Additionally, some evaluation variables for the map-matching result are presented to quantify persisting inaccuracies due to the chosen network type and the behaviour of some vehicles.

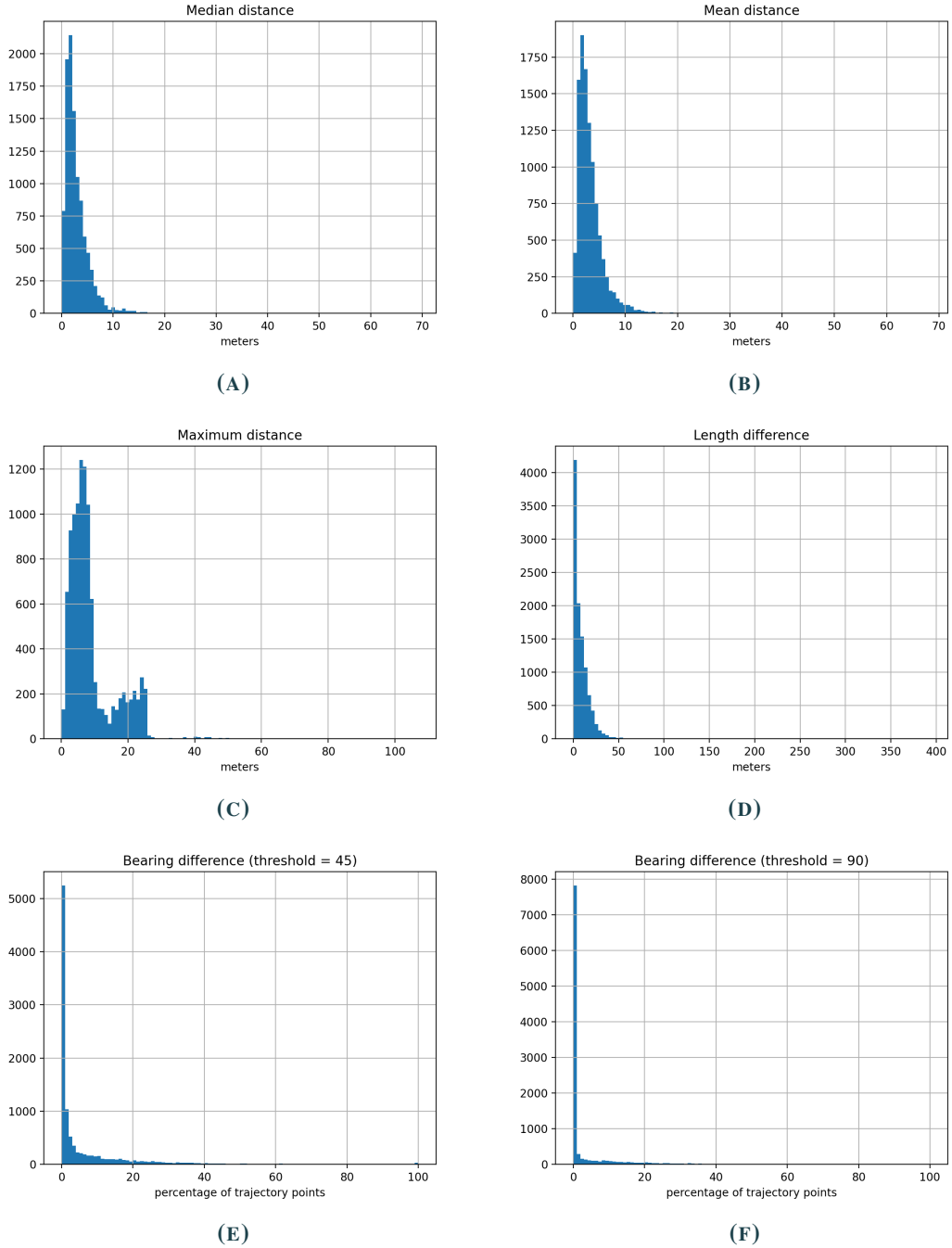

FIGURE 3.11
Histograms of variables used for evaluation: A, B and C show the median, mean and maximum straight-line distance to the matched edge, respectively, D. difference between trajectory and matched length, E and F. percentage of trajectory observations with bearing difference exceeding 45 and 90 degrees, respectively

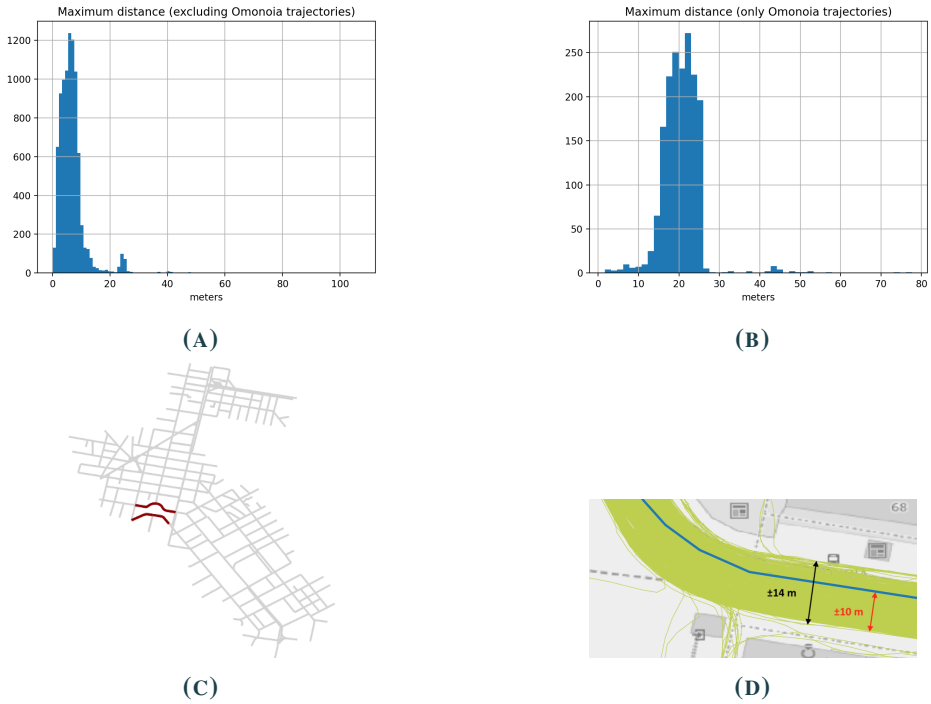

FIGURE 3.12
Maximum distance histograms: A. Trajectories not travelling on Omonoia edges B. Only trajectories travelling on Omonoia edges C. Edges of Omonoia square D. Width in black and maximum distance in red of all trajectories matched to an edge

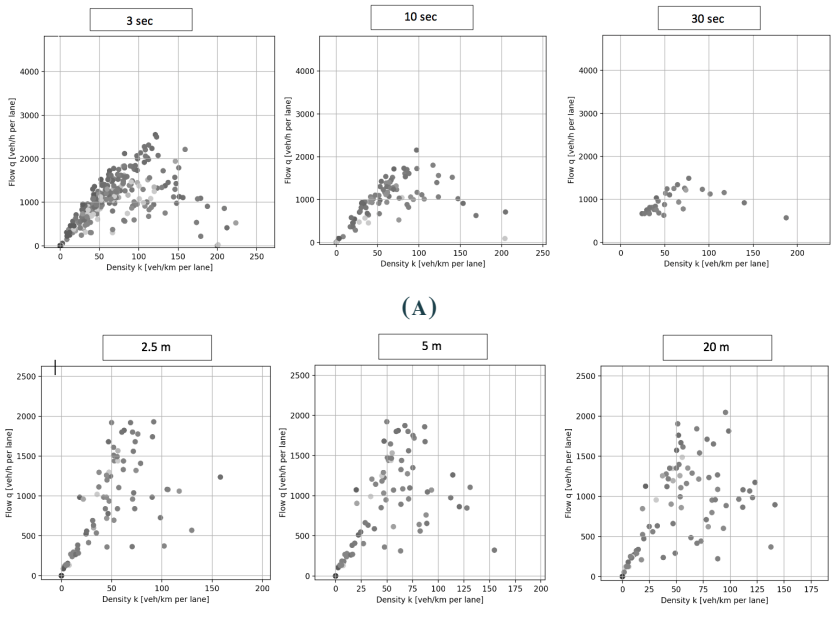

FIGURE 3.13
Space-time area: A. Flow-density scatter plots for different ΔT B. Flow-density scatter plots for different ΔX

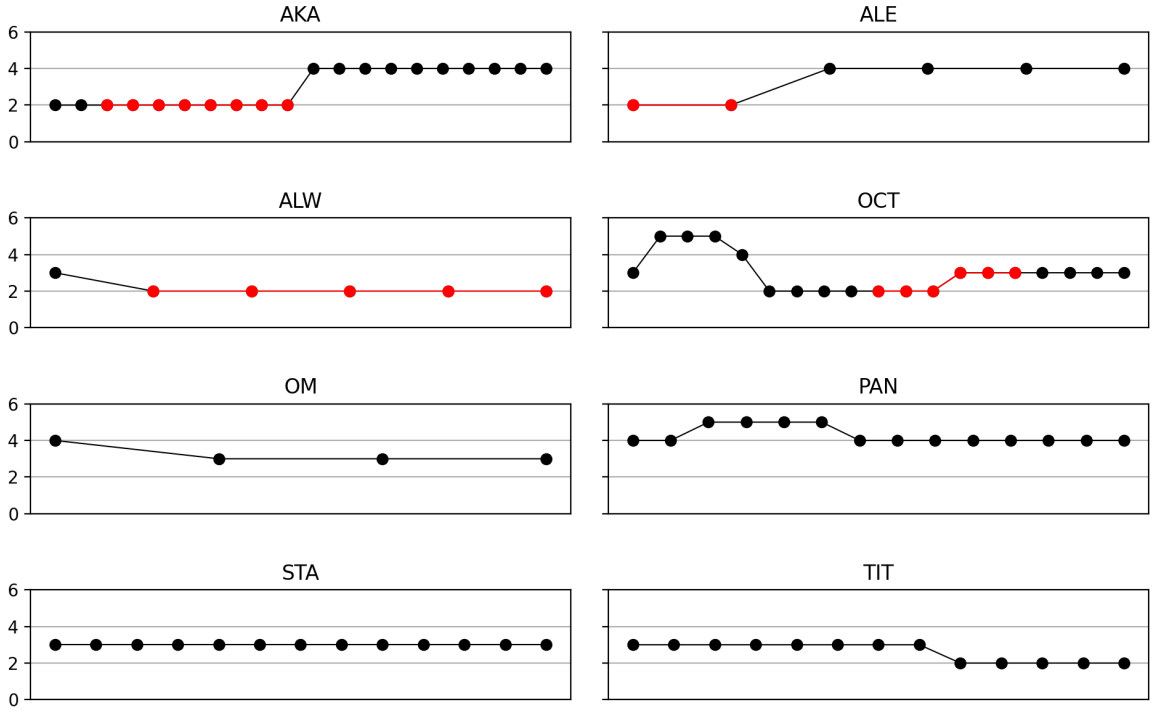

FIGURE A.1
Number of lanes for every arterial with DBL sections marked in red

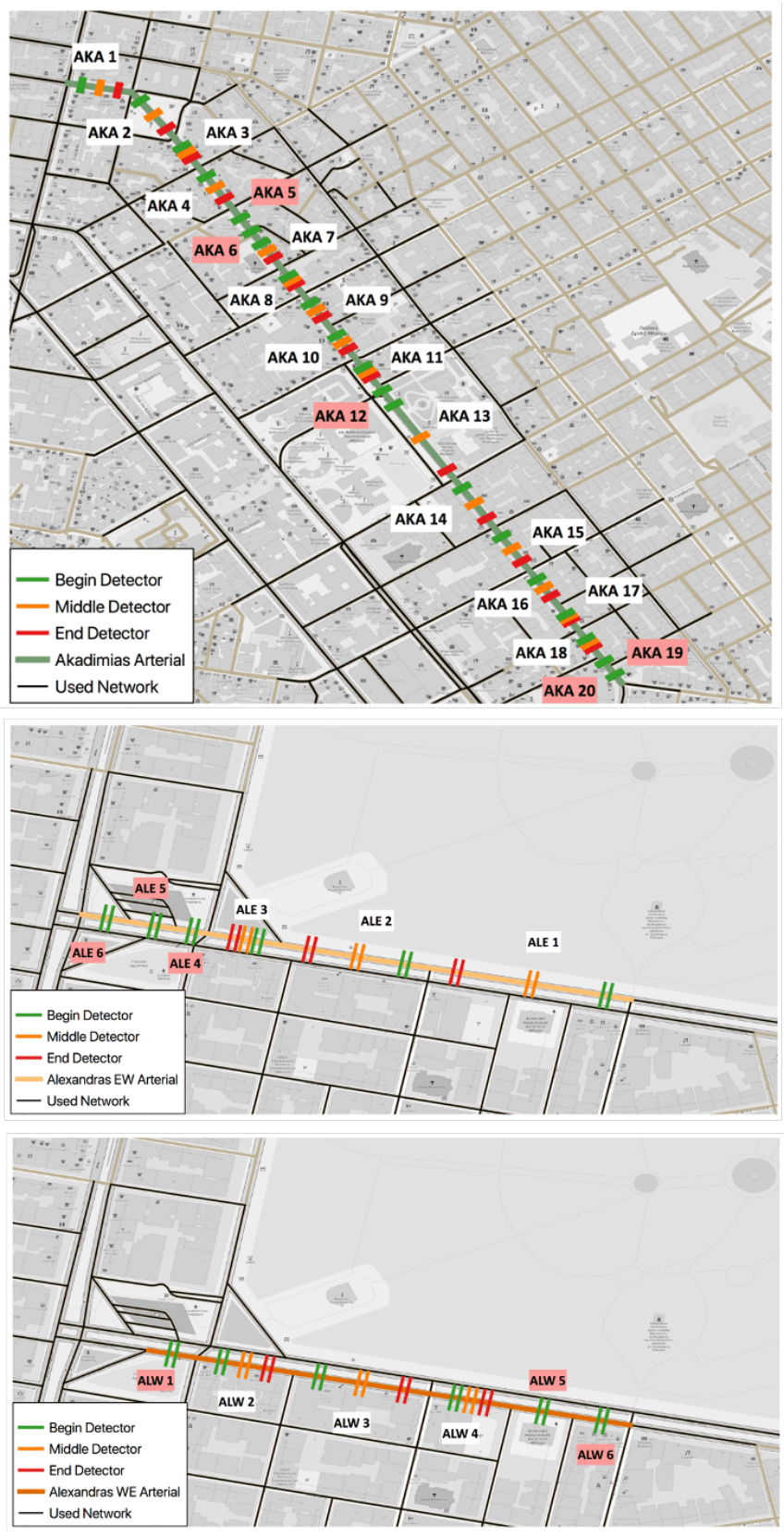
AKA 1
AKA 2
AKA 3
AKA 4
AKA 5
AKA 6
AKA 7
AKA 8
AKA 9
AKA 10
AKA 11
AKA 12
AKA 13
AKA 14
AKA 15
AKA 16
AKA 17
AKA 18
AKA 19
AKA 20
Begin Detector
Middle Detector
End Detector
Akadimias Arterial
Used Network
ALE 5
ALE 3
ALE 2
ALE 1
ALE 6
ALE 4
Begin Detector
Middle Detector
End Detector
Alexandras EW Arterial
Used Network
ALW 1
ALW 2
ALW 3
ALW 4
ALW 5
ALW 6
Begin Detector
Middle Detector
End Detector
Alexandras WE Arterial
Used Network

FIGURE A.2

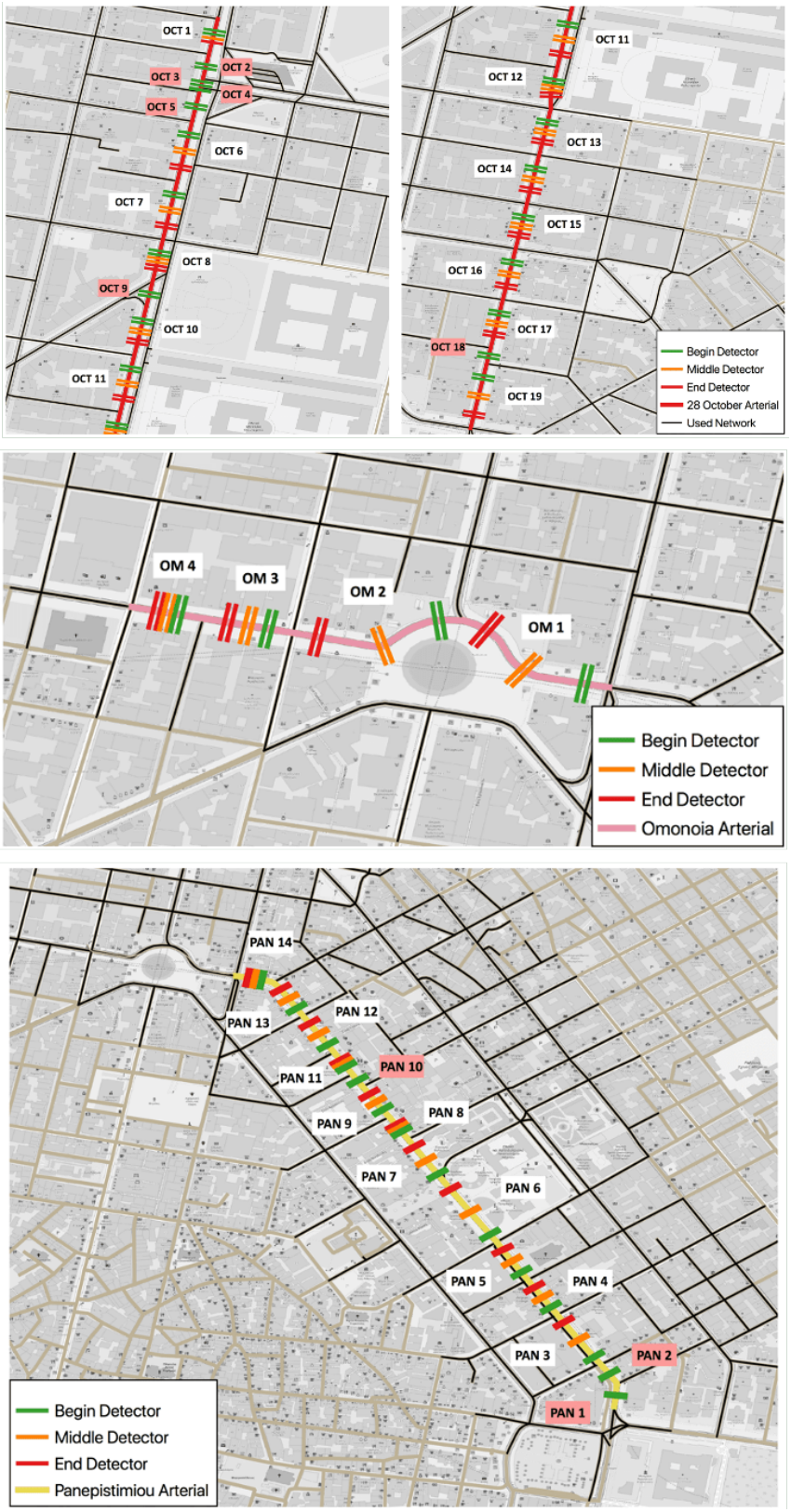
OCT 1
OCT 2
OCT 3
OCT 4
OCT 5
OCT 6
OCT 7
OCT 8
OCT 9
OCT 10
OCT 11
OCT 12
OCT 13
OCT 14
OCT 15
OCT 16
OCT 17
OCT 18
OCT 19
Begin Detector
Middle Detector
End Detector
28 October Arterial
Used Network
OM 4
OM 3
OM 2
OM 1
Begin Detector
Middle Detector
End Detector
Omonoia Arterial
PAN 14
PAN 13
PAN 12
PAN 11
PAN 10
PAN 9
PAN 8
PAN 7
PAN 6
PAN 5
PAN 4
PAN 3
PAN 2
PAN 1
Begin Detector
Middle Detector
End Detector
Panepistimiou Arterial

FIGURE A.3

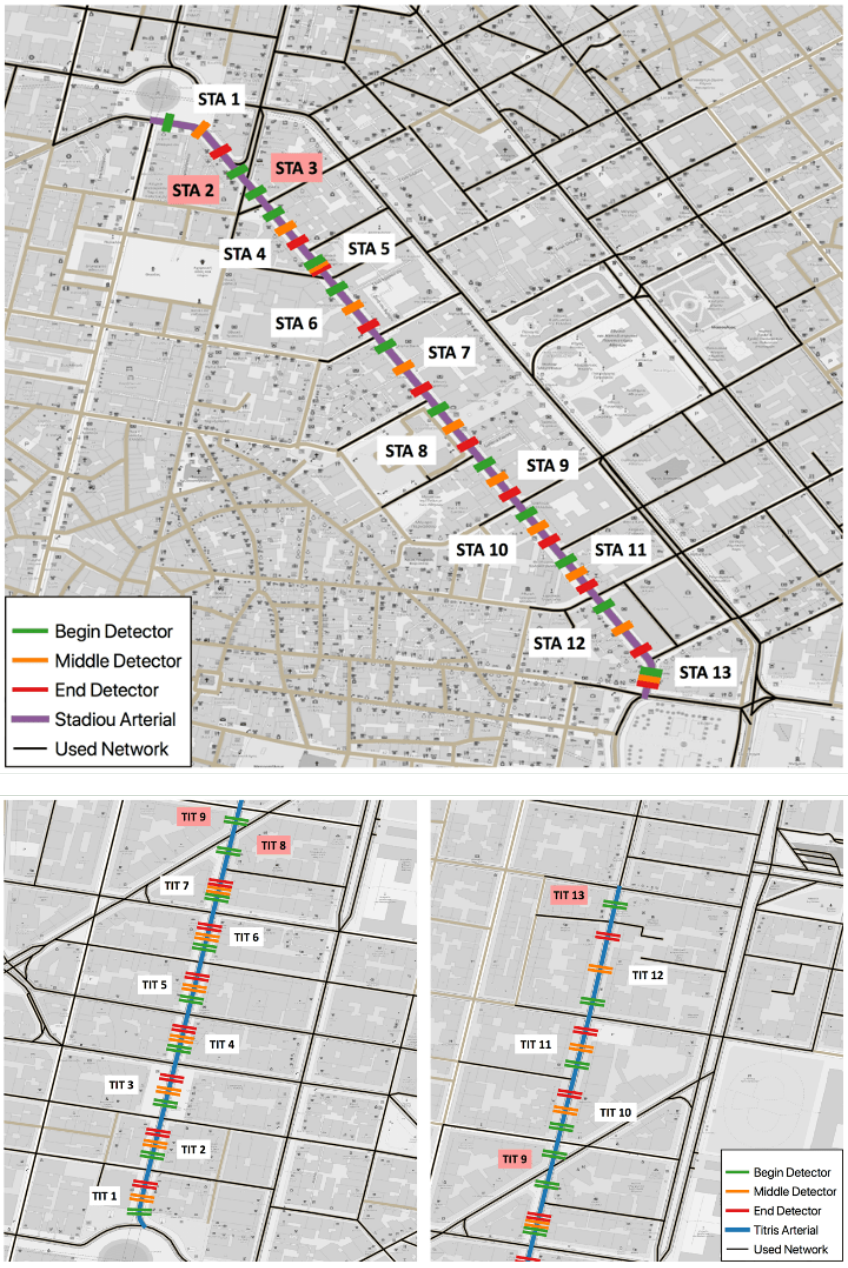
STA 1
STA 2
STA 3
STA 4
STA 5
STA 6
STA 7
STA 8
STA 9
STA 10
STA 11
STA 12
STA 13
Begin Detector
Middle Detector
End Detector
Stadiou Arterial
Used Network
TIT 9
TIT 8
TIT 7
TIT 6
TIT 5
TIT 4
TIT 3
TIT 2
TIT 1
TIT 13
TIT 12
TIT 11
TIT 10
TIT 9
Begin Detector
Middle Detector
End Detector
Titris Arterial
Used Network

FIGURE A.4